Luis Manuel Pereira da Costa

Licenciado em Engenharia Informática e de Computadores

Desenvolvimento e Aplicação do Algoritmo Enxame de Partículas na Determinação da Máxima Injeção Nodal em Redes de Energia Elétrica

Dissertação para obtenção do Grau de Mestre em
Energias Renováveis – Conversão Elétrica e Utilização Sustentável

Orientador: Professor Doutor Francisco Alexandre Ganho da Silva Reis, Professor, Faculdade de Ciência e Tecnologia, da Universidade Nova de Lisboa

Co-orientador: Professor Doutor Mário Fernando da Silva Ventim Neves, Professor, Faculdade de Ciência e Tecnologia, da Universidade Nova de Lisboa

Júri:

Presidente: Prof. Doutor Fernando José Almeida Vieira do Coito

Arguente: Prof. Doutor Pedro Miguel Pereira

Vgais: Prof. Doutor Francisco Alexandre Ganho da Silva Reis

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2014

Desenvolvimento e Aplicação do Algoritmo Enxame de Partículas na Determinação da Máxima Injeção Nodal em Redes de Energia Elétrica

Aos meus pais,

À minha esposa,

Aos meus irmãos,

Às minhas sobrinhas.

Agradecimentos

Agradeço à Faculdade de Ciência e Tecnologia da Universidade Nova de Lisboa, a oportunidade de realização desta dissertação, a todos os docentes e colegas que me apoiaram ao longo deste curso de mestrado em Energias Renováveis.

Expresso o meu agradecimento especial ao meu orientador da tese, Professor Doutor Francisco Alexandre Ganho da Silva Reis, pelo seu encorajamento, disponibilidade e conhecimentos transmitidos e ao coordenador do mestrado, Professor Doutor Mário Ventim Neves, que fizeram com que fosse possível a sua realização.

Um especial agradecimento à minha esposa Júlia e à minha irmã Cristina, pela compreensão, paciência e apoio que me têm dado ao longo destes anos.

Resumo

O objetivo desta dissertação é a determinação da máxima injeção nodal numa rede de energia elétrica, ou seja, qual o valor total máximo de potência ativa que é possível injetar e qual a sua distribuição pelos diversos nós da rede simultaneamente. Determinámos esta máxima injeção nodal em duas situações distintas: injeção não simultânea, injetando potência em um só nó de cada vez e injeção simultânea, injetando potência em todos os nós da rede simultaneamente.

Sendo este um problema de natureza combinatória, utilizámos para esta determinação o algoritmo conhecido como nuvem ou enxame de partículas, adaptando-o ao nosso problema. Desenvolvemos o software na linguagem de programação *Python* utilizando o ambiente *Eclipse*. Para resolver o trânsito de energia utilizámos o programa *PSSE University.*

Para os exemplos de aplicação utilizámos duas redes de energia elétrica, uma de 6 e outra de 14 barramentos. Estas redes foram baseadas nas redes *IEEE 6 BUS* e *IEEE 14 BUS* respetivamente.

Concluímos que o algoritmo nuvem ou enxame de partículas cumpriu o objetivo traçado, obtendo as melhores soluções para cada um dos casos, máxima injeção nodal não simultânea e máxima injeção nodal simultânea. No contexto deste problema, o parâmetro chave do algoritmo, comprovado pelos ensaios feitos, é a velocidade máxima de deslocação das partículas, tomando valores típicos de 7 a 10 para a rede de 6 barramentos e de 20 a 25 para a de 14 barramentos.

Palavras-chave: máxima injeção nodal, injeção nodal não simultânea, injeção nodal simultânea, algoritmo enxame de partículas, trânsito de energia.

Abstract

The goal of this dissertation is the determination of the maximum nodal injection in an electrical power grid, in other words, what is the maximum total active power it is possible to inject in a power grid and which is the distribution of that power for the several grid buses simultaneously. We have determined that maximum nodal power injection in two different situations: non simultaneous injection, injecting power in one bus at a time and simultaneous injection, injecting power in all the grid buses simultaneously.

Being this a combinatory problem, we have used for this determination, the algorithm known as *Particle Swarm optimization*, adapting it to our problem. To develop the software we have used the *Python* programming language together with the *Eclipse* programming platform. For the power flow we have used the *PSSE university* program.

To test the algorithm we have used two electrical power grids one of 6 and the other of 14 buses. These two power grids were based upon the *IEEE 6 BUS* and *IEEE 14 BUS* electrical test power grids, respectively.

We have concluded that the *Particle Swarm optimization* algorithm has carried out the goal drawn, obtaining the best solution for each one of these two cases, non simultaneous and simultaneous injection. In the context of this problem, the key parameter of the algorithm, proved by the essays done, is the maximum speed of the particle, taking typical values from 7 to 10 for the 6 buses grid and from 20 to 25 for the 14 buses grid.

Keywords: maximum nodal injection, non simultaneous nodal injection, simultaneous nodal injection, particle swarm optimization, power flow.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

API ..... Application Program Interface

C_1 ..... Fator Cognitivo Individual

C_2 ..... Fator Cognitivo Social

G_{best} ..... Global best

IEEE ..... Institute of Electrical and Electronic Engineers

IEEE 6 BUS ..... Rede elétrica de teste, de 6 barramentos, do IEEE

IEEE 10 BUS ..... Rede elétrica de teste, de 10 barramentos, do IEEE

Iter ..... Iteração corrente

$Iter_{max}$ ..... Número máximo de iterações

$Iter_{min}$ ..... Número mínimo de iterações

L_{best} ..... Local best

P_{best} ..... Particle best

P_{max} ..... Potência máxima

P_{min} ..... Potência mínima

PSO ..... Particle Swarm Optimization

r_1, r_2 ..... Funções aleatórias no intervalo [0,1]

v ..... Velocidade da partícula

v_{max} ..... Velocidade máxima da partícula

v_{min} ..... Velocidade mínima da partícula

w ..... Fator de inércia

w_{max} ..... Valor máximo do fator de inércia

w_{min} ..... Valor mínimo do fator de inércia

x ..... Posição da partícula

x_{max}, x_{min} ..... Posições máxima e mínima da partícula, que delimitam o espaço de busca

w ..... Fator de inércia

1

Primeiro Capítulo - Introdução

1.1. Motivação

A necessidade de estudar o problema da injeção nodal apareceu quando o paradigma da geração e distribuição de energia elétrica começou a mudar. Em vez de uma geração de energia centralizada em grandes centrais, uma rede de transporte para a transportar até junto dos consumidores e uma rede de distribuição para a distribuir por estes, temos hoje e muito devido à crescente integração das energias renováveis, centrais de menores dimensões em locais propícios ao modo de geração de cada uma, sendo que estas injeções de energia ocorrem em geral nos nós da rede de distribuição.

Devido à especificidade da geração de energias renováveis, são criadas centrais com menor potência instalada, em locais determinados por fatores como por exemplo, o vento no caso da energia eólica (os aerogeradores são instalados onde há vento e este é favorável à geração de energia eólica).

Esta mudança de paradigma não se deve só a esta especificidade das energias renováveis. Estando a geração, qualquer que ela seja, mais perto do consumo, as perdas de energia devido ao transporte são menores.

Se se souber quais os nós da rede mais favoráveis à instalação da nova geração, ou seja, aqueles que maximizam o total de potência que esta consegue absorver, pode-se injetar mais potência na rede minimizando os reforços necessários para acomodar este acréscimo de energia.

1.2. Objetivos

Com este trabalho pretendeu-se estudar o problema da máxima injeção nodal, de potência ativa, numa rede de energia elétrica, de modo a serem determinados quais os melhores nós para injetar esta potência e qual o seu valor máximo.

Pretendeu-se adaptar um algoritmo, que será o *PSO* (Particle Swarm Optimization), ao caso da injeção nodal, simultânea e não simultânea e codificá-lo, de modo a poder ser aplicado à generalidade das redes.

Pretendeu-se ainda deixar sugestões para futuros desenvolvimentos deste trabalho.

1.3. Estrutura

Esta dissertação está dividida em cinco capítulos.

No capítulo um, apresenta-se o problema, descrevem-se os objetivos deste estudo, a motivação para estudar este problema e a encontrar uma solução para ele e a estrutura da dissertação.

No capítulo dois formula-se o problema, fala-se de estudos que tenham já sido feitos sobre ele, da sua natureza e das potenciais formas de resolução.

No capítulo três introduz-se o algoritmo escolhido para a resolução do problema, adapta-se ao caso em estudo e implementa-se para o caso das injeções não simultâneas e simultâneas.

No capítulo quatro apresenta-se a codificação dos algoritmos e a sua aplicação às redes de 6 e 14 barramentos.

O capítulo cinco destina-se às observações finais e às perspetivas de desenvolvimento futuro.

2

Segundo Capítulo - O Problema da Máxima Injeção Nodal

2.1. Formulação do problema

Como já foi dito anteriormente, neste estudo propôs-se encontrar uma solução, aplicável à generalidade das redes, para maximizar a potência absorvida por estas. Este problema tem duas vertentes, uma em que se injeta potência em cada nó da rede separadamente, para se determinar qual a potência máxima que se pode injetar em cada nó, um de cada vez e outra em que se injeta potência em todos os nós da rede simultaneamente, para se determinar qual a potência máxima que se pode injetar em cada nó, injetando em todos ao mesmo tempo. A seguir vamos formulam-se estes dois casos separadamente.

2.1.1. Injeções não simultâneas

Na determinação da potência máxima em cada nó, considerando injeção não simultânea, para uma determinada rede elétrica já existente, injeta-se potência num nó, ficando todos os outros com a geração que já lá existia. Aumentando a potência até uma linha entrar em sobrecarga ou o gerador do nó de referência estar a gerar uma potência igual a zero, consegue-se determinar qual a potência máxima injetada nesse nó, estando todos os outros, exceto o de referência, no seu estado inicial, em termos de geração. Pode-se então formular o problema do seguinte modo:

Considerando um nó genérico i, a função objetivo (2.1) será a maximização da potência nesse nó i:

$$f(P_{Gi}) = \max \{P_{Gi}\} \qquad (2.1)$$

onde P_{Gi} é a potência ativa gerada no nó i.

com i = 1, 2, ..., n nós e $P_G = P_{G1}, P_{G2}, ..., P_{Gi}$

e sujeita às restrições:

A potência ativa e a tensão em cada nó estão compreendidas entre um máximo e um mínimo.

$$P_{min,i} \leq P_i \leq P_{max,i}, \quad i = 1, ..., n \text{ nós} \qquad (2.2)$$

$$V_{min,i} \leq V_i \leq V_{max,I}, \quad i = 1, ..., n \text{ nós} \qquad (2.3)$$

Não pode haver linhas em sobrecarga.

$$S_k \leq S_{maxk}, \quad k = 1, ..., n \text{ linhas} \qquad (2.4)$$

A potência ativa gerada no nó de referência não pode ser inferior a zero

$$P_{GRef} \geq 0 \qquad (2.5)$$

Na figura 2.1 apresenta-se um exemplo de uma rede com 6 barramentos onde se injetam 150 MW de potência ativa no barramento 6, que inicialmente não tinha geração, deixando todos os outros barramentos com a geração já existente. Os barramentos 1, 2 e 3 já tinham geração de potência ativa (440, 260 e 240 MW respetivamente e os barramentos 4 e 5 não tinham geração). Neste caso, aumentar-se-ia a potência do gerador do barramento 6 até um valor onde uma restrição (2.2), (2.3), (2.4) ou (2.5) fosse violada. O valor imediatamente anterior seria o valor de potência que poderia ser injetada no nó 6, desde que a geração de potência em todos os outros ficasse inalterada (à exceção do nó de balanço).

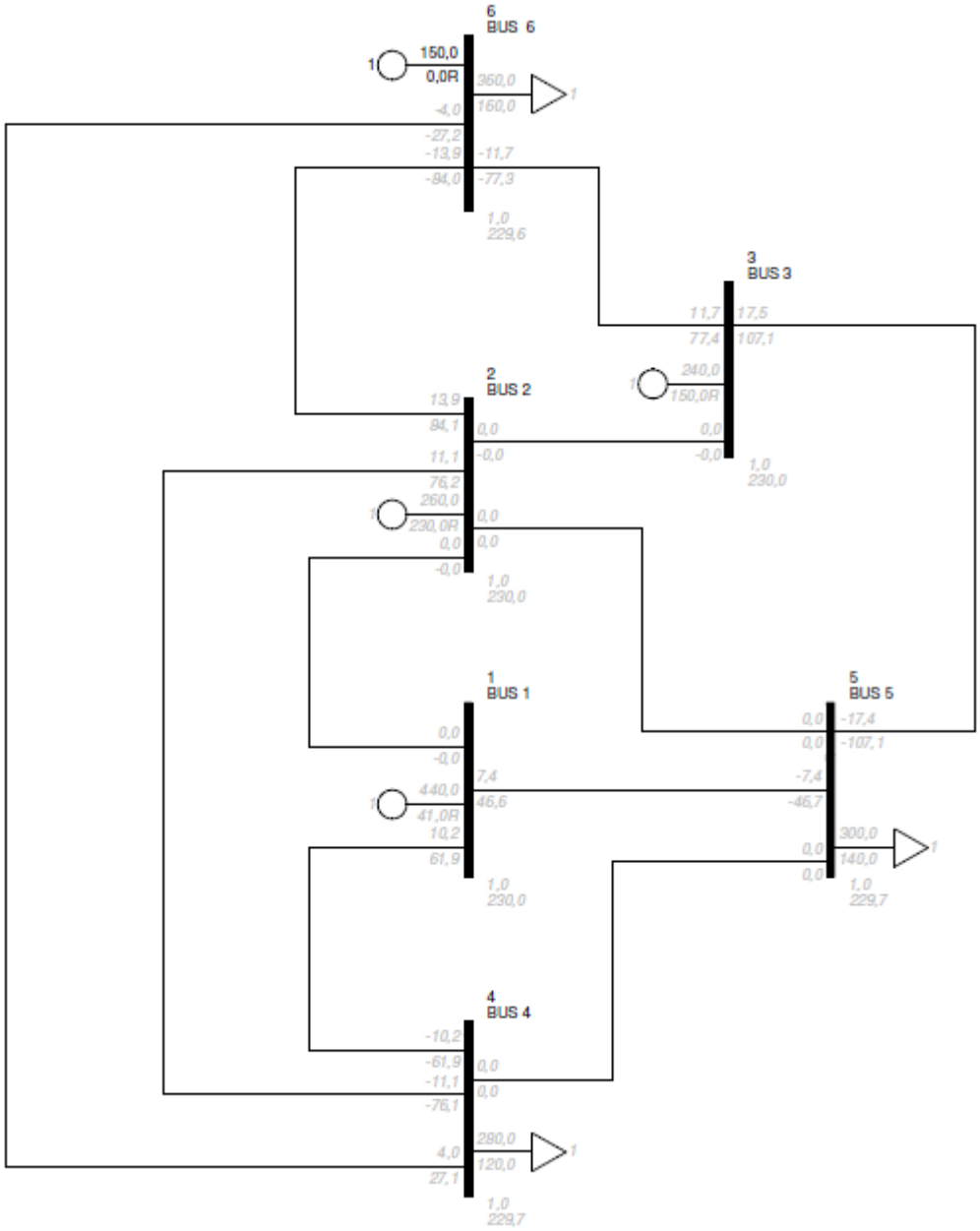


Figura 2.1: Rede de 6 barramentos com injeção de 150 MW de potência ativa no nó 6, ficando todos os outros com a geração original

2.1.2. Injeções simultâneas

Considerando a injeção simultânea, que é o problema que é proposto ser resolvido, a potência a maximizar será o somatório das potências injetadas em cada nó. O problema a resolver será encontrar a

combinação de potências a injetar nos nós de modo a tornar essa potência máxima. Então, a função objetivo pode ser representada pela expressão (2.6) e sujeita às restrições (2.2), (2.3), (2.4) e (2.5) apresentadas no parágrafo 2.1.1:

$$f(P) = \max\left\{\sum_{i=1}^{n} P_{Gi}\right\} \tag{2.6}$$

onde P_{Gi} é a potência a injetar no nó i.

com i = 1, 2, ..., n nós e $P_G = P_{G1}, P_{G2}, ..., P_{Gi}$

Na figura 2.2 apresenta-se um exemplo de uma rede com 6 barramentos onde é injetada potência ativa em todos os nós. Nos que já tinham geração, não se altera a já existente, instalando um segundo gerador para injetar o acréscimo de potência. Foi o que aconteceu nos barramentos 1, 2 e 3 onde já existia geração de potência ativa (440, 260 e 240 MW respetivamente). Neste caso, a potência total injetada na rede, valor que se quer otimizar, é o somatório das potências injetadas em cada nó, depois de resolvido o trânsito de energia. No fim, o que se pretende determinar é a combinação de valores de potência ativa que se deve ter em cada um dos geradores, os que foram acrescentados, de forma a ter-se um valor máximo que respeite as restrições impostas no parágrafo 2.1.1.

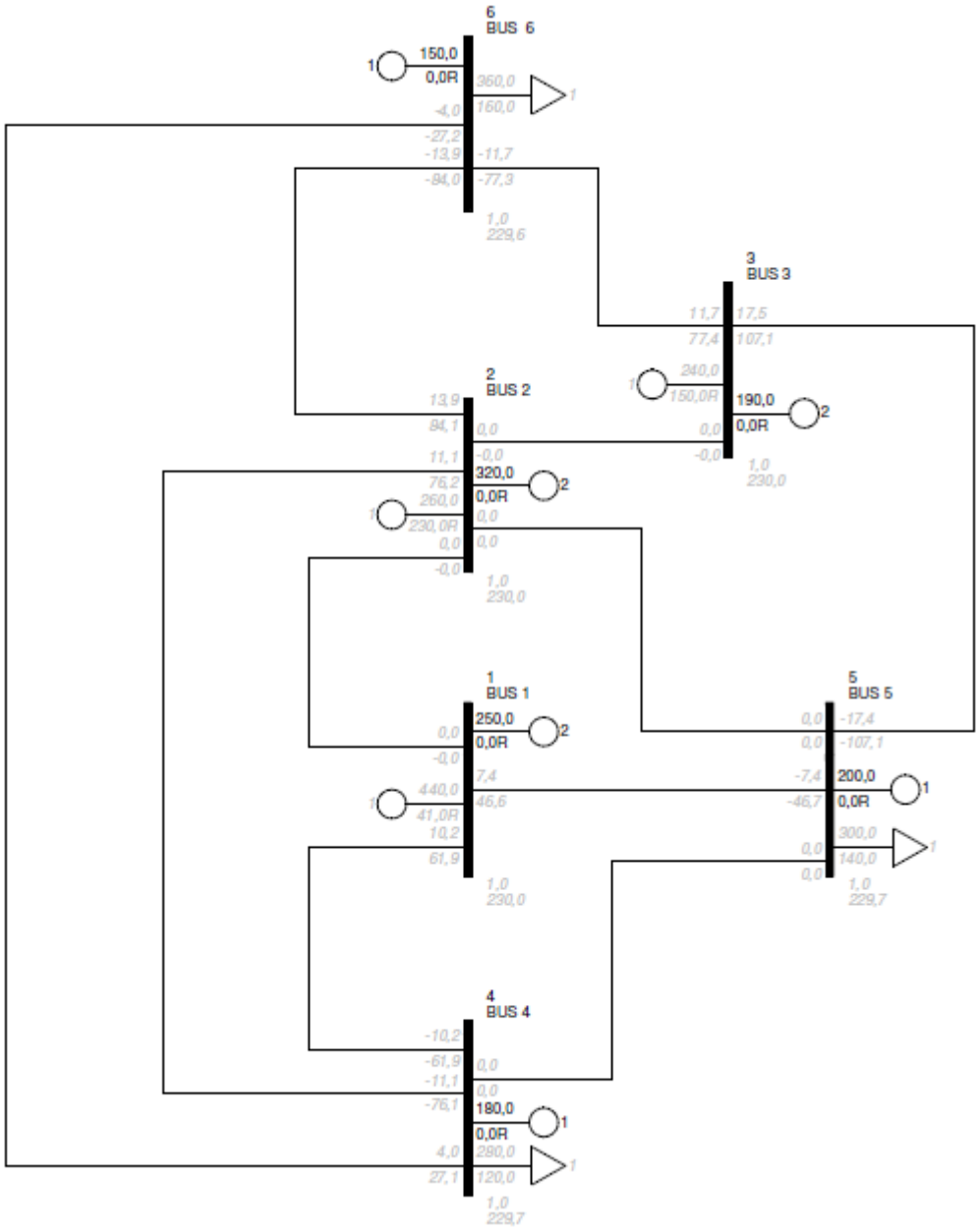


Figura 2.2: Rede de 6 barramentos com injeção de potência ativa em todos os nós, mantendo a geração original

2.2. Estudos já realizados sobre o problema

Na pesquisa feita, foram identificados outros estudos sobre este tema, utilizando, na sua maioria, algoritmos genéticos para endereçar o problema da máxima injeção nodal.

Alguns autores desenvolveram técnicas para localizar a geração distribuída de modo a minimizar as perdas na rede, como é o caso de Raw e Wan [1], que usam gradiente e métodos de segunda ordem para determinar os locais ótimos de injeção de modo a minimizar as perdas e as cargas na rede e Kim *et all* [2], que recorre a uma combinação de programação difusa e técnicas de algoritmos genéticos para localizar os nós de injeção e minimizar as perdas totais. Harrison e Wallace [3], propõem uma metodologia denominada "*método da carga inversa*", onde cada novo gerador é visto como uma carga negativa e fazem o despacho com o OPF (*Optimal Power Flow*), pretendendo assim identificar os nós onde devem ser instalados os novos geradores e maximizar a potência a instalar na rede.

Em [4], João Nunes propõe algoritmos genéticos para resolver este problema. Tal como no presente, o seu trabalho tem como objetivo estabelecer a capacidade máxima de energia que é possível injetar na rede, identificando os nós mais favoráveis à receção da nova energia.

Bezaliel Pires [5] propõe o algoritmo PSO (*Partcle Swarm Optimization*) para determinação dos valores máximos de potência ativa a serem injetados em nós pré-determinados das redes a estudar. Este estudo tem a restrição de as novas perdas não poderem ser superiores às que existiam no caso de não haver geração distribuída.

2.3. Natureza do problema

O problema da máxima injeção nodal levanta duas questões. Por um lado, quer-se saber qual a potência máxima que podemos aplicar

num determinado nó, por outro, como esse valor é afetado com a injeção de potência nos outros nós da rede, para serem respeitadas as restrições físicas da rede elétrica e não se baixar a qualidade do serviço (limites mínimos e máximos de tensão e carga das linhas).

Querendo injetar potência ativa num único nó, resolve-se o trânsito de energia na rede e vai-se injetando, nesse nó, valores de potência cada vez maiores até se determinar o valor máximo a instalar. Querendo injetar em dois ou mais nós, já vai ter de ser resolvido o trânsito de energia para as outras combinações possíveis de valores em cada um dos nós pretendidos para se poder determinar a melhor combinação, aquela que maximiza ou minimiza a função objetivo, respeitando as restrições impostas pelo problema. No presente caso, quer-se determinar o valor máximo de potência a injetar na rede, pretende-se maximizar a função objetivo (2.1) apresentada no parágrafo 2.1.

Quanto maior for o número de nós da rede maior será o número de combinações possíveis de valores em cada nó. Este é portanto, um problema de natureza combinatória.

2.4. Potenciais formas de resolução

Um problema combinatório é um problema onde existem várias combinações possíveis de soluções. De entre essas combinações possíveis, uma será a melhor solução para o problema. As possíveis soluções do problema podem ser obtidas usando o método de carga inversa, proposto por Harrison e Wallace em [3], algoritmos genéticos, como

no trabalho de João Nunes, em [4], métodos probabilísticos, como propõe Jorge de Almeida em [6], …

Foi escolhido o algoritmo *PSO* (Particle Swarm Optimization) ou otimização por enxame de partículas, em português, para encontrar as combinações possíveis de solução para o problema. Este algoritmo foi escolhido por ser um dos algoritmos possíveis para resolver o problema em questão, por fazer parte do grupo de algoritmos da chamada inteligência artificial que deriva de uma forma de inteligência coletiva e por ter sido, na pesquisa efetuada, encontrado apenas um estudo que aplica este algoritmo a este problema, mas impondo a restrição de as novas perdas não poderem ser superiores às que existiam no caso de não haver geração distribuída. No presente caso, essa restrição não é imposta porque o objetivo é determinar a máxima energia que a rede pode absorver independentemente das perdas.

3

Terceiro Capítulo – Adaptação do Algoritmo *PSO* ao Problema da Máxima Injeção Nodal

3.1. Introdução

Este algoritmo, *PSO* (Otimização por Enxame ou Nuvem de Partículas), que foi escolhido para resolver o problema da máxima injeção nodal, é baseado no comportamento e nas vantagens que certos indivíduos possuem ao viver em grupo, como um enxame de abelhas, um cardume de peixes, um bando de pássaros... Neste texto, os termos enxame ou nuvem são utilizados genericamente para referência de qualquer grupo de agentes capazes de interagir entre si. Estes agentes, muitas vezes pouco inteligentes ou desprovidos mesmo de inteligência, mostram, quando trabalham em conjunto, comportamentos coletivos inteligentes. A estes comportamentos, White e Pagurek [7] deram o nome de inteligência de enxame.

3.1.1. Apresentação do Algoritmo

As origens deste algoritmo remontam a 1990 quando o biólogo Frank Heppner [8] publicou um estudo onde descreveu um modelo do comportamento coletivo de bandos de pássaros. Neste modelo, os pássaros são distribuídos no espaço, aleatoriamente, para começarem a procurar alimento e um local para fazer o ninho. No início, eles voam, também aleatoriamente, seguindo apenas o seu instinto. Quando um pássaro encontra alimento ou local para o ninho, os outros pássaros percebem isso, principalmente os que estão na sua vizinhança e dirigem-se para esse local. É a informação de um indivíduo a ser partilhada pelo coletivo. Há medida que mais pássaros vão encontrando alimento e locais, vão-se formando vários grupos dentro do bando, onde cada pássaro partilha a sua informação com os seus vizinhos. O local onde houver um maior número de pássaros, será tido como a localização onde há mais alimento ou mais favorável para fazer o ninho, ou as duas coisas. Este comportamento mostra uma cooperação entre os pássaros do bando. Cada pássaro, que inicialmente só tinha o seu instinto para seguir e depois o seu conhecimento, adquirido na procura, com este comportamento social, começou a poder contar não só com o melhor conhecimento dos pássaros da sua vizinhança mas também com o melhor conhecimento de todo o bando.

Em 1995, o psicólogo James Kennedy e o engenheiro eletrotécnico Russel Eberhart [9], desenvolveram um algoritmo de otimização baseado no estudo de Frank Heppner. Neste algoritmo, o *PSO*, cada indivíduo é chamado de partícula e é uma solução possível para o pro-

blema. Cada uma das partículas que formam a nuvem ou enxame, desloca-se num plano de *d* dimensões à procura da solução ideal, como os pássaros se deslocam à procura de alimento e local para o ninho. À semelhança dos pássaros, o deslocamento de cada partícula é feito tendo em conta a melhor posição encontrada até ao momento por ela mesma, P_{best} (Particle best), a melhor posição encontrada por partículas da sua vizinhança, L_{best} (Local best) e a melhor posição encontrada por quaisquer partículas do enxame, G_{best} (Global best). Após o deslocamento, as partículas são avaliadas e a sua velocidade e posição são alteradas de acordo com os parâmetros já enunciados. Este ciclo repete-se até que uma ou mais partículas tenham satisfeito o critério de paragem ou o número máximo de iterações tenha sido atingido.

Ainda à semelhança dos pássaros, um aspeto importante deste algoritmo é a vizinhança e o conceito de vizinhança. Uma partícula pode receber informações apenas das partículas que lhe estão mais próximas ou pode receber de todas as partículas, ou de ambos. Este conceito de vizinhança será analisado mais adiante. A escolha da topologia de vizinhança está relacionada com o tipo de problema a resolver e vai influenciar a maneira como as partículas exploram o espaço.

Na figura 3.1 apresenta-se o fluxograma genérico do algoritmo *PSO*.

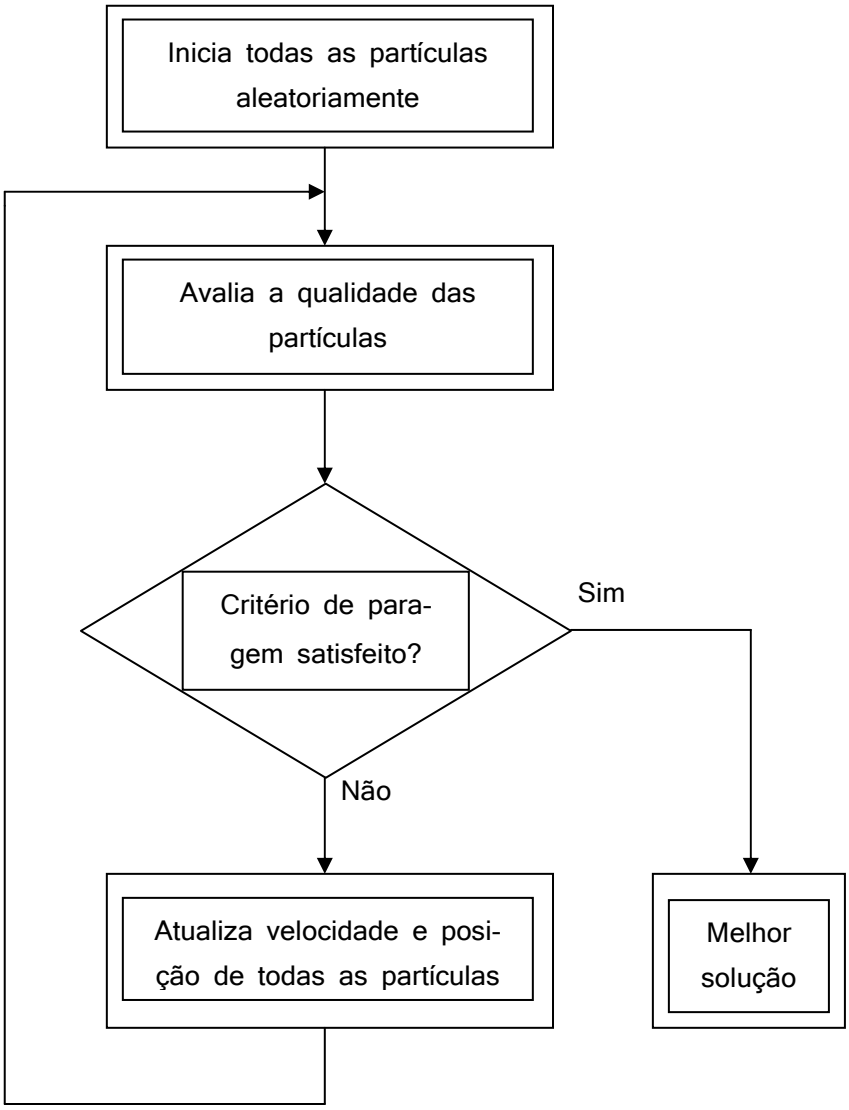


Figura 3.1: Fluxograma genérico do algoritmo *PSO*

Quando o algoritmo *PSO* desenvolvido por Kennedy e Herberhart, foi apresentado, foi dito que no seu deslocamento a partícula se serve do conhecimento armazenado por si própria, pelas partículas da sua vizinhança e por todas as partículas. A seguir ver-se-ão as diversas topologias de vizinhança onde uma partícula pode estar inserida, de acordo com Mateus Rosendo [10]:

1. **Vazia**. Neste tipo de vizinhança a partícula não está ligada a mais nenhuma outra a não ser ela própria. Quer dizer que no deslocamento ela só vai poder contar com o conhecimento armazenado por ela própria, P_{best}.

2. **Melhor local**. Por oposição a *Melhor global*. A partícula está ligada a *n* partículas. Dessas n partículas uma detém a melhor posição e é essa que vai influenciar todas as outras dessa vizinhança. Quando *n=2*, esta topologia é conhecida como *topologia em anel* (figura 3.2). Uma partícula está ligada só a outras duas. Estas vizinhanças são dinâmicas. No seu movimento, ao afastar-se das partículas a que está ligada e aproximar-se de outras, qualquer partícula pode mudar de vizinhança. No seu movimento, a partícula utiliza o seu conhecimento, P_{best} e o conhecimento da partícula melhor posicionada, na sua vizinhança, L_{best}.

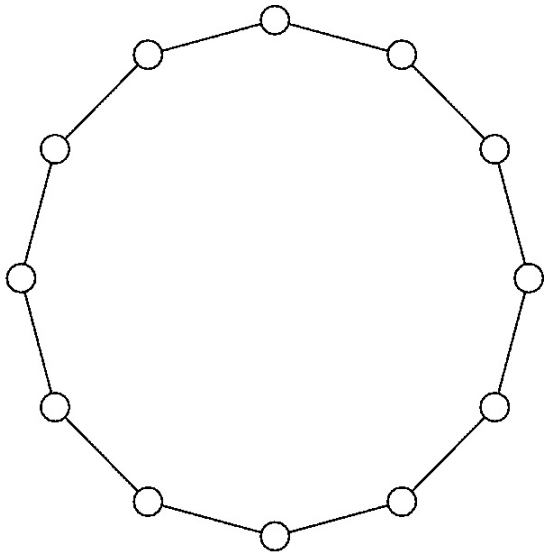


Figura 3.2: Topologia em Anel. Fonte [10]

3. **Em estrela**. Todas as partículas estão ligadas a uma só partícula habitualmente conhecida como partícula focal (figura 3.3). Esta partícula compara a posição de todas as partículas e movimenta-se de acordo com a melhor posicionada. As restantes partículas movimentam-se de acordo com a partícula focal. Deste modo, a informação da melhor partícula é rapidamente propagada às restantes, embora estas recebam sempre, também a influência da partícula focal.

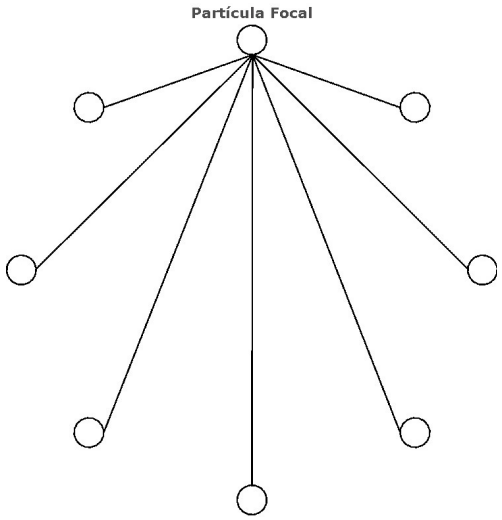


Figura 3.3: Topologia em Estrela. Fonte [10]

4. **Em árvore**. Nesta topologia as partículas são organizadas em árvore (figura 3.4). Cada nó *pai* tem dois nós *filhos* e cada nó *pai* influencia dois nós *filhos*. Só os *pais* influenciam os *filhos* e nunca o contrário, porque os *pais* têm sempre melhores posições (no espaço de busca) que os *filhos*. Se um filho encontra uma posição melhor do que a do seu pai, esses dois nós trocam de posição na árvore. À semelhança da topologia *Melhor local* (2.), no seu movimento, a partícula utiliza o seu conheci-

mento, P_{best} e o conhecimento da partícula melhor posicionada, na sua vizinhança, L_{best}.

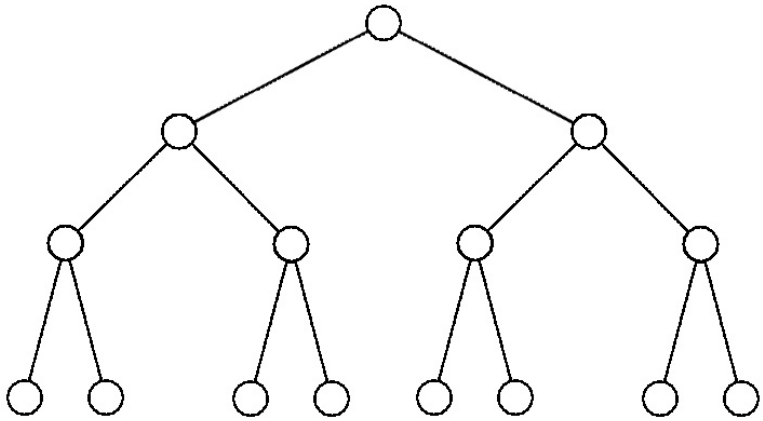


Figura 3.4: Topologia em Árvore. Fonte [10]

5. **Completamente ligado**. Por oposição à topologia da vizinhança vazia (1.), nesta, qualquer partícula está ligada com qualquer outra (figura 3.5). Quer isto dizer, que além do seu conhecimento, qualquer partícula pode usar o conhecimento, P_{best} e o da partícula melhor posicionada de todas, no espaço de busca, G_{best}. Isto significa que a vizinhança foi alargada a todo o espaço e deste modo, $L_{best} = G_{best}$.

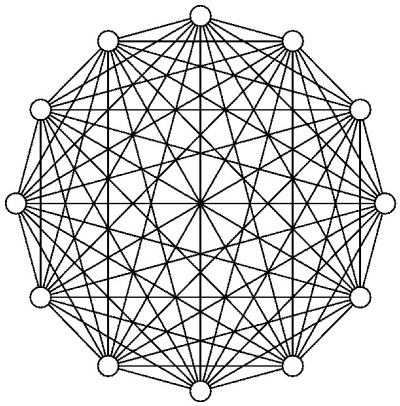


Figura 3.5: Topologia Completamente Ligado. Fonte [10]

3.1.2. Conceito de partícula

A partícula é a pedra base do algoritmo *PSO*. O número *d*, de dimensões do plano, depende do problema a resolver. No caso do problema que se quer resolver, será o número de barramentos da rede a analisar.

Como já foi dito, cada partícula é uma solução possível para o problema. Não existe um controlo central que comanda as partículas. São elas que ao longo do processo vão aprendendo o caminho até ao objetivo servindo-se de toda a informação, quer a aprendida por cada uma, quer a disponibilizada por todas as outras.

No caso deste problema, máxima injeção nodal, a partícula terá em cada dimensão o valor de potência a injetar em cada nó da rede. Tomando como exemplo a aplicação deste algoritmo a uma rede de 4 barramentos para determinar a máxima injeção nodal simultânea, tendo a restrição de a potência a injetar em cada nó não poder ser inferior a 0 nem superior a 50 MW, um exemplo de partícula é apresentado na tabela 3.6:

Tabela 3.1: Exemplo de partícula

15 MW	28 MW	17 MW	42 MW
Barramento 1	Barramento 2	Barramento 3	Barramento 4

A partícula pode ser uma estrutura de dados tão elaborada quanto se quiser, dependendo da informação que se quer que ela contenha. Se se quiser que a partícula armazene mais informação, por exemplo a

melhor combinação de potências que encontrou até agora (P_{best}), o somatório dessas potências (potência absorvida pela rede com aquela configuração, que é o que se quer maximizar) e a melhor configuração encontrada até ao momento, teremos uma partícula como a da figura 3.6.

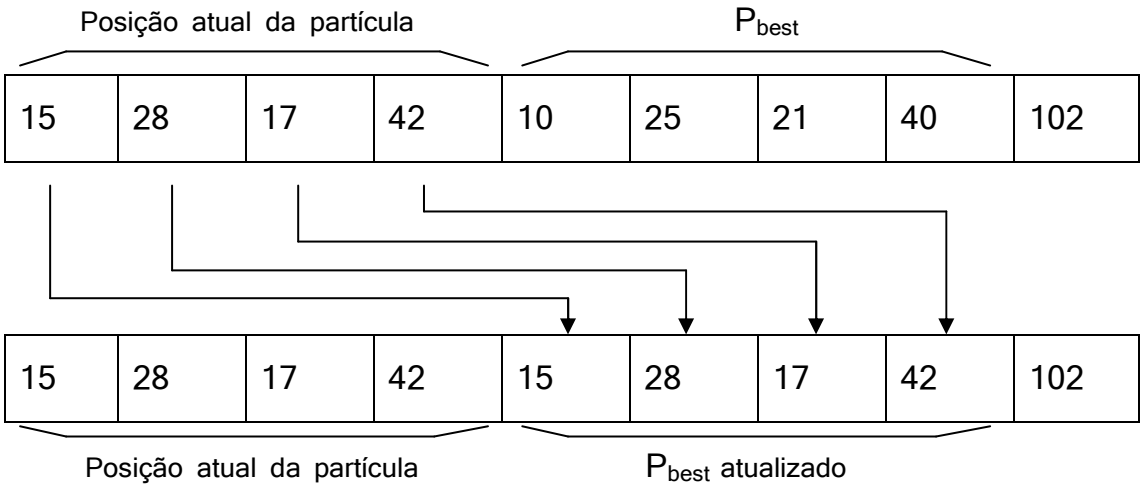


Figura 3.6: Outro exemplo de partícula, onde o P_{best} é atualizado

Os primeiros 4 membros são a posição atual da partícula, os 4 seguintes são a melhor posição que a partícula tinha encontrado até então, P_{best} e o último membro é o somatório das potências nodais, que se quer maximizar. Neste exemplo, a posição atual é melhor do que P_{best}, o somatório é 102 MW em vez de 96, então P_{best} seria atualizado. Os membros 10, 25, 21 e 40 seriam afetados com os membros 15, 28, 17 e 42 respetivamente, porque deixariam de representar a melhor posição encontrada até ao momento, ou seja P_{best}. Este passaria agora a ser representado pela posição "15, 28, 17 e 42". Em cada iteração a velocidade e a posição de cada partícula são atualizadas. Ao

fim de um determinado número de iterações as partículas começam a convergir para a melhor solução. Este processo é geralmente conhecido por curva de aprendizagem das partículas. É apresentada na figura 3.7 um exemplo de uma curva de aprendizagem do algoritmo *PSO*.

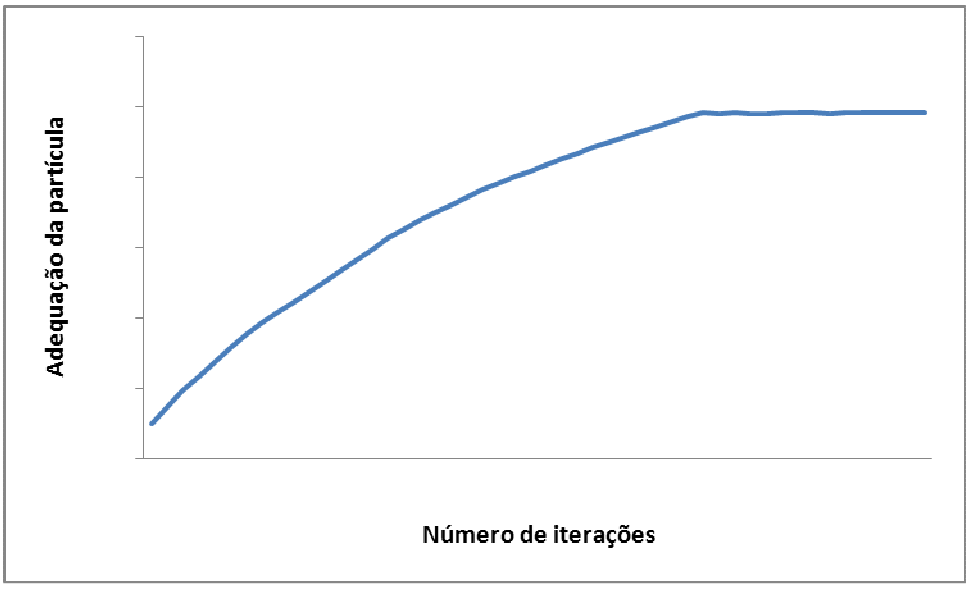


Figura 3.7: Exemplo de curva de aprendizagem da partícula no algoritmo *PSO*

3.1.3. Parametrização

A nova velocidade da partícula é determinada pela equação (3.1) e é condicionada por 3 fatores[(1)]:

1. Fator de inércia, *w*, determina a influência que a velocidade atual da partícula vai ter na próxima velocidade.

---

(1) Neste estudo foi escolhido o modelo de PSO em que é alargada a vizinhança da partícula a todo o espaço de busca, ficando $L_{best} = G_{best}$.

2. Fator cognitivo individual, que determina a influência que a melhor posição encontrada pela partícula, até ao momento, P_{best}, vai ter na próxima velocidade.
3. Fator cognitivo social, que determina a influência que a melhor posição encontrada, entre todas as partículas, até ao momento, G_{best}, vai ter na próxima velocidade.

$$v(t+1) = w*v(t) + c_1 * r_1 * (P_{best}(t)-x(t)) + c_2 * r_2 * (G_{best}(t)-x(t)) \quad (3.1)$$

Parâmetros da equação (3.1):

1. ***w***, o fator de inércia, como já mencionado, é o que representa aqui a autoconfiança da partícula. Um alto valor para *w*, leva a partícula a seguir mais o seu caminho e menos o caminho determinado pelas melhores posições, tanto a dela própria, P_{best}, como a melhor de todas as partículas G_{best}. Valores baixos, de *w*, incentivam uma procura local, enquanto valores elevados incentivam uma procura global. De acordo com E. R. C. Viveros [11] e M. A. Abido [12], se o fator de inércia for ajustado para um valor alto no início, fará com que as partículas se espalhem mais pelo espaço na procura da solução. À medida que as partículas se vão agrupando e convergindo para a melhor solução, o valor de *w*, vai sendo diminuído gradualmente para que haja aí uma exploração mais intensiva do espaço, tendo mais em conta o conhecimento adquirido pela partícula e pelo enxame e menos a "intuição" da partícula. Porque nesta fase, tanto a

partícula como o enxame onde ela está integrada já adquiriram conhecimento suficiente para saberem o caminho a seguir. Este comportamento pode ser modulado com a equação (3.2).

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} * iter \qquad (3.2)$$

Dependendo do número máximo de iterações, $iter_{\max}$, no início o valor de w será perto do máximo, $w_{\max}$. Se considerarmos $iter_{\max} >> w_{\max} - w_{\min}$, o que normalmente acontece, pode-se dizer que $w \backsimeq w_{\max}$. No último ciclo, quando $iter = iter_{\max}$, $w = w_{\min}$. Ao ser escolhido um fator de inércia dinâmico, modulado pela equação (3.2), a velocidade da partícula também vai variar, sendo mais elevada no início e baixando gradualmente com o número de iterações pois, como se verá, a inércia vai ter influência na velocidade da partícula.

2. $\mathbf{c_1}$ e $\mathbf{c_2}$ são duas constantes inteiras, positivas e correspondem às componentes cognitivas individual e coletiva ou social, respetivamente. Estas constantes são geralmente ajustadas no intervalo contínuo [0,2], [9] ao afinar-se o algoritmo depois de codificado. Estes coeficientes determinam o peso que P_{best} e G_{best} têm no movimento da partícula, ou seja, o quanto a partícula confia na sua própria solução (componente cognitiva individual) ou na do enxame (componente cognitiva social). Na

maioria dos casos, os valores destes fatores são iguais. Neste estudo optou-se por fazer $c_1 = c_2 = 1{,}49618$ [13].

3. $\mathbf{r_1}$ e $\mathbf{r_2}$ são duas funções aleatórias no intervalo contínuo [0,1]. Estes dois parâmetros também colaboram no peso atribuído às componentes cognitivas. Vão fazer variar o peso dessas duas componentes, de uma forma aleatória, mantendo assim a diversidade da população. Deste modo as partículas distribuem a sua atenção pelos seus resultados e pelos resultados do grupo.

4. $\mathbf{P_{best}}$ (Particle best), é a melhor posição encontrada pela partícula até ao momento. Este parâmetro tem influência na atualização da velocidade da partícula como se pode ver na equação (3.1). O peso deste parâmetro depende da função aleatória r_1 e do fator de cognição individual, c_1.

5. $\mathbf{G_{best}}$ (Global best), é a melhor posição encontrada entre todas as partículas. Este parâmetro tem influência na atualização da velocidade da partícula como se pode ver na equação (3.1). O peso deste parâmetro depende da função aleatória r_2 e do fator de cognição social, c_2.

6. **V(t)** é a velocidade da partícula na iteração t.

7. **V(t+1)** é a velocidade da partícula na iteração seguinte (t+1).

8. **X(t)** é a posição da partícula na iteração t.

Número de partículas (parâmetro do algoritmo). Este parâmetro influencia claramente o comportamento do algoritmo, pois quanto maior for o número de partículas num determinado espaço de busca, maior será a probabilidade de ser encontrada a melhor solução ou uma solução perto desta.

Número de iterações (parâmetro do algoritmo). Quanto maior for o número de iterações maior será também a probabilidade de ser encontrada a melhor solução ou uma solução perto desta, já que as partículas têm mais espaço para explorar e portanto, mais hipóteses de se aproximarem da melhor solução.

Uma vez determinada a nova velocidade da partícula, pela equação (3.1), a sua nova posição é atualizada pela equação (3.3).

$$x(t+1) = x(t) + v(t+1) \tag{3.3}$$

Quanto maior for o número de partículas e o número de iterações, maior é a probabilidade de o algoritmo encontrar a melhor solução para o problema. Por outro lado, um maior número de partículas e iterações, vai resultar em maior número de testes e atualizações e consequentemente em maior tempo de computação. Devido a isto, todos os parâmetros devem ser criteriosamente escolhidos conforme o problema a resolver.

O espaço de busca é normalmente delimitado por uma posição máxima $X(t)_{max}$ e uma posição mínima $X(T)_{min}$. Estas posições limites

são escolhidas de acordo com o problema. No caso presente, serão as potências máxima e mínima determinadas para cada nó. Não nos interessa ter as partículas a explorar espaços onde já sabemos não se encontrarem soluções válidas para o problema.

À semelhança do espaço, a velocidade também é limitada por uma velocidade máxima $V(t)_{max}$ e por uma velocidade mínima $V(T)_{min}$. Uma velocidade alta pode fazer as partículas aproximarem-se mais rapidamente do patamar onde deverá estar a solução ótima, havendo uma certa probabilidade de deixar para trás regiões do plano por explorar. Nessas regiões podem estar máximos ou mínimos locais relevantes para a solução do problema. Uma velocidade baixa requer mais iterações para as partículas começarem a convergir no patamar, mas aumenta a probabilidade de uma maior exploração do espaço de busca.

Para cada problema, os números de partículas e de iterações, os limites do espaço de busca e os limites de velocidade, devem ser escolhidos de acordo com a especificidade desse problema, de modo a que ao fim de um determinado número de iterações em que as partículas cooperam na procura da solução, a curva de aprendizagem tenda a estabilizar, como se pode ver na figura 3.8, estando nesse patamar a melhor solução do problema. Esse patamar deve também ser entendido como um dos critérios de paragem.

Na figura 3.8 exemplificou-se o movimento de uma partícula no espaço bidimensional. Neste exemplo, pode-se dizer que a partícula e o enxame ainda não adquiriram conhecimento suficiente para terem a certeza do caminho a seguir. Estando a falar de um algoritmo com fator

de inércia dinâmico, pode ver-se que nesta fase a inércia ainda tem um peso significativo e a partícula sofre um desvio apreciável na sua trajetória. Pode ver-se na figura que o vetor que resulta da multiplicação da velocidade com o fator de inércia (w∗V(t)) é da ordem de grandeza dos outros dois vetores que também determinam o deslocamento da partícula.

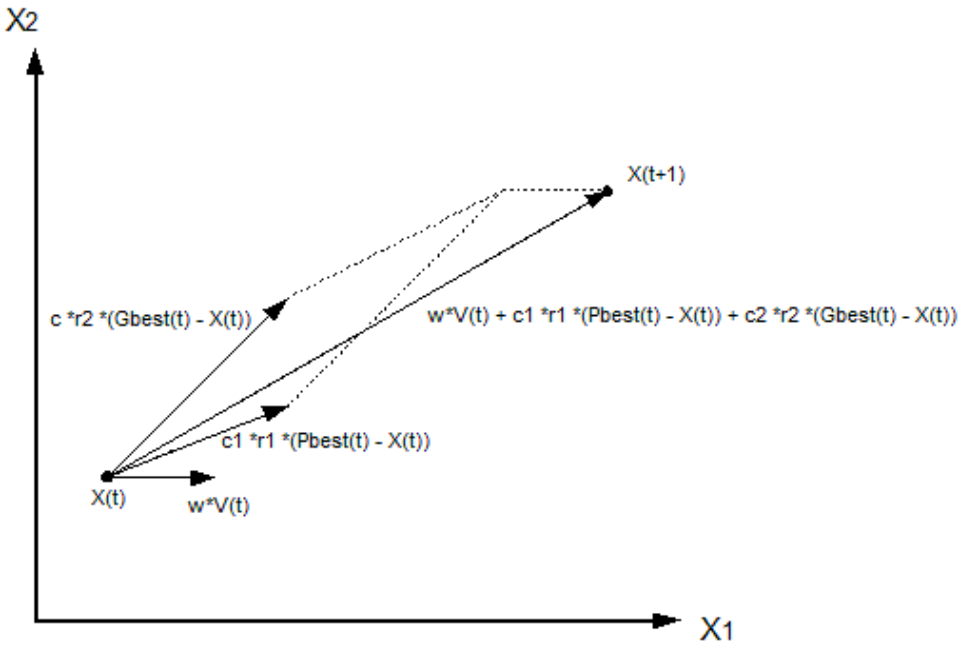


Figura 3.8: Exemplo do movimento de uma partícula no espaço bidimensional

3.2. **Avaliação da partícula**

Pode dizer-se que a partícula é a peça fulcral do algoritmo PSO. Como se viu, é nela que vão sendo armazenados os valores de potência ativa a injetar nos nós. Cada partícula representa portanto, um caso de rede de energia elétrica. Sempre que uma partícula é gerada ou atualizada, tem de se verificar se essa partícula é válida ou não. Para isso,

aplica-se essa partícula (valores de potência ativa) à rede e resolve-se o trânsito de energia com ela, para se ver se alguma restrição foi violada ou não. Perante o resultado do trânsito de energia o algoritmo decidirá que classificação dar à partícula. De seguida será visto um método para resolver o trânsito de energia numa rede.

Basicamente, uma rede de energia elétrica é composta por nós, chamados também barramentos e ramos, que são as linhas incluindo os transformadores, os geradores e as cargas. A energia transita nos ramos da rede de uns nós para os outros. Genericamente pode falar-se em sistemas de energia (grupo de uma ou mais redes). Nestes sistemas o número de nós e de ramos é normalmente muito elevado, da ordem de centenas ou milhares e as equações que os modelam são não lineares. Por isso, para resolver o trânsito de energia, também chamado trânsito de potência (Power Flow)[(1)], nestes sistemas, é necessário um meio de cálculo muito potente [14].

Começa-se por formular um modelo matemático que represente, em regime estacionário, o sistema real que queremos analisar. Este modelo deve ser o mais rigoroso possível. De seguida especifica-se o tipo de barramentos e as grandezas associadas a cada um. O próximo passo é a determinação das equações do trânsito de energia. A solução numérica

---

(1) *Power Flow* é a designação em língua inglesa para o cálculo do trânsito de energia na rede. Determina os módulos e argumentos das tensões nos nós da rede, a potência injetada no nó de balanço e as potências que transitam nas linhas.

destas equações fornece os módulos e os argumentos das tensões nos nós. Finalmente calculam-se as potências que transitam nos ramos (linhas e transformadores) e as potências geradas e consumidas nos barramentos, conforme o seu tipo.

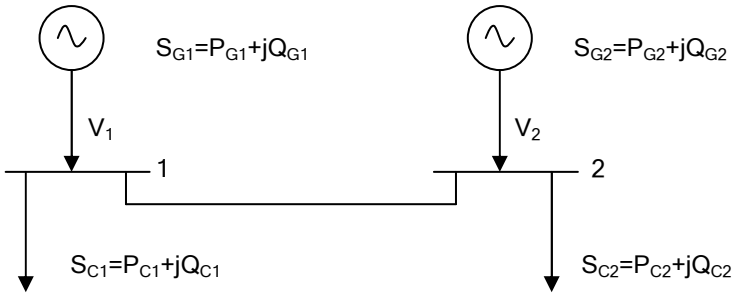
Existem 3 tipos de barramentos, conforme especificado na tabela 3.2.

Tabela 3.2: Tipos de barramentos

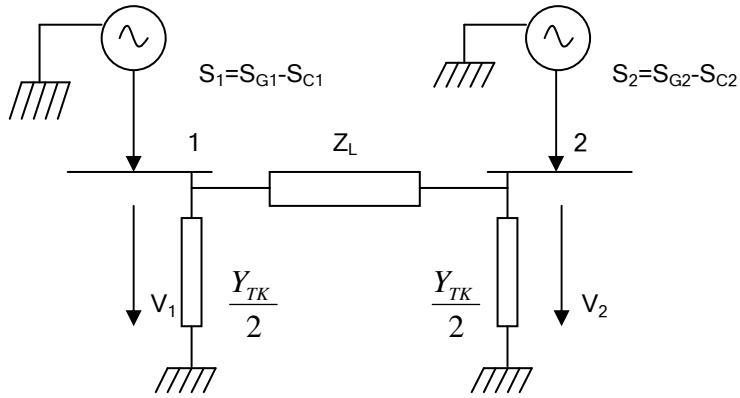
Tipo de barramento	Variáveis conhecidas	Variáveis especificadas	Variáveis calculadas
Referência/balanço	P_C Q_C	$\|V\|$ θ	P_G Q_G
PQ (carga ou geração)	P_C Q_C	P_G Q_G	$\|V\|$ θ
PV (geração)	P_C Q_C	P_G $\|V\|$	Q_G θ

3.2.1. Rede de dois barramentos

Vai-se agora determinar as equações do trânsito de energia para o caso mais simples. A rede de dois barramentos (figura 3.9). Em b) a rede está representada pelo seu esquema equivalente em Π: uma impedância longitudinal e duas admitâncias transversais, uma em cada extremo da linha. Ainda em b) os geradores fictícios injetam nos nós uma potência igual à diferença da potência gerada pelo gerador do nó com a potência consumida pela carga presente nesse nó (definição de potência injetada num nó).



a)



b)

Figura 3.9: Rede de dois barramentos
a) Esquema unifilar;
b) Esquema equivalente em π

Esta rede é constituída por:

- Barramento 1, que tem ligado um gerador que fornece uma potência complexa S_{G1} e S_{G2} e uma carga S_{C1}.
- Barramento 2, que tem ligado um gerador que fornece uma potência complexa S_{G2} e uma carga S_{C2}.

- Uma linha de transmissão a ligar os dois barramentos. Esta linha, no seu modelo equivalente em π, é representada por uma impedância longitudinal Z_L e uma admitância transversal $Y_T/2$ em cada extremo da linha (figura 3.1b).

A potência injetada num barramento é a diferença entre as potências gerada e consumida nesse barramento. Então, para os barramentos 1 e 2 temos:

$$S_1 = P_1 + jQ_1 = P_{G1} - P_{C1} + j(Q_{G1} - Q_{C1}) \tag{3.4}$$

$$S_2 = P_2 + jQ_2 = P_{G2} - P_{C2} + j(Q_{G2} - Q_{C2}) \tag{3.5}$$

Quando num barramento a potência injetada é positiva, significa que a potência gerada é superior à consumida pela carga. Quer dizer que esse barramento está a injetar energia na linha. Quando é negativa, o barramento consome energia que lhe chega através da linha. Se for zero, significa que toda a energia gerada naquele barramento é ali consumida. Sabendo que

$$I = \frac{S^*}{V^*} = \frac{P - jQ}{V^*} \tag{3.6}$$

Aplicando a lei dos nós a ambos os barramentos:

$$I_1 = \frac{S_1^*}{V_1^*} = \frac{Y_T}{2}V_1 + \frac{1}{Z_L}(V_1 - V_2) \tag{3.7}$$

$$I_2 = \frac{S_2^*}{V_2} = \frac{Y_T}{2}V_2 + \frac{1}{Z_L}(V_2 - V_1) \tag{3.8}$$

Definindo $y_{11} = y_{22} = \frac{Y_T}{2} + \frac{1}{Z_L}$ e $y_{12} = y_{21} = -\frac{1}{Z_L}$ (3.9) e (3.10)

vem $\frac{S_1^*}{V_1^*} = y_{11}V_1 + y_{12}V_2$ (3.11)

e $\frac{S_2^*}{V_2^*} = y_{21}V_1 + y_{22}V_2$ (3.12)

De (3.6), (3.11) e (3.12), temos que:

$$P_1 - jQ_1 = y_{11}V_1V_1^* + y_{12}V_2V_1^* \quad (3.13)$$

$$P_2 - jQ_2 = y_{21}V_1V_2^* + y_{22}V_2V_2^* \quad (3.14)$$

Ou, em notação compacta,

$$P_i - jQ_i = V_i^* \sum_{j=1}^{2} y_{ij}V_j \text{ , i = 1, 2} \quad (3.15)$$

As equações (3.15), que representam (3.13) e (3.14) escritas em notação compacta, são as equações para calcular o trânsito de energia na rede de dois barramentos (figura 3.1). Estas equações estão na forma complexa. Portanto, pode-se separar as suas componentes reais e imaginárias obtendo assim quatro equações.

Sabendo que a tensão complexa em coordenadas polares é:

$$V_i = v_i e^{j\theta_i} \text{ , i = 1, 2} \quad (3.16)$$

E que a admitância complexa em coordenadas retangulares é:

$$y_{ij} = G_{ij} + jB_{ij} \text{ , i = 1, 2} \quad (3.17)$$

onde G_{ij} e B_{ij} são a condutância e a suscetância nodais, respetivamente. Substituindo (3.16) e (3.17) em (3.15), obtem-se, também em notação compacta:

$$P_i - jQ_i = \sum_{j=1}^{2} (G_{ij} + jB_{ij}) v_i v_j e^{j(\theta_i - \theta_j)} \text{ , i = 1, 2} \quad (3.18)$$

Separando agora as partes real e imaginária:

$$P_i = P_{Gi} - P_{Ci} = \sum_{j=1}^{2} v_i v_j \left[G_{ij} \cos(\theta_i - \theta_j) + B_{ij} sen(\theta_i - \theta_j) \right], \text{ i = 1, 2} \qquad (3.19)$$

$$Q_i = Q_{Gi} - Q_{Ci} = \sum_{j=1}^{2} v_i v_j \left[G_{ij} sen(\theta_i - \theta_j) - B_{ij} \cos(\theta_i - \theta_j) \right], \text{ i = 1, 2} \qquad (3.20)$$

Chega-se a um sistema de quatro equações algébricas, (3.19) e (3.20) escritas em notação compacta, porque modelam o sistema em regime estacionário. Estas equações não são lineares e nestes casos usam-se métodos iterativos.

Numa rede, a potência gerada pelos geradores é igual à potência consumida pelas cargas mais a potência perdida nos ramos (linhas e transformadores). Isto é válido tanto para a potência ativa como para a reativa. É o balanço de potência. Então, se somando as equações (3.19), obtem-se a equação do balanço de potência ativa. Dizendo que $P_L = P_1 + P_2$ representa as perdas na rede, temos:

$$P_{G1} + P_{G2} = P_{C1} + P_{C2} + P_L \qquad (3.21)$$

$$P_L = (v_1^2 + v_2^2) G_{11} + 2 v_1 v_2 G_{12} \cos(\theta_1 - \theta_2) \qquad (3.22)$$

Fazendo o mesmo para as duas equações (3.20):

$$Q_{G1} + Q_{G2} = Q_{C1} + Q_{C2} + Q_L \qquad (3.23)$$

$$Q_L = -(v_1^2 + v_2^2) B_{11} - 2 v_1 v_2 B_{12} \cos(\theta_1 - \theta_2) \qquad (3.24)$$

Ao contrário das perdas de potência ativa que são sempre positivas, as de potência reativa podem ser positivas ou negativas. Olhando para as equações (3.22) e (3.24), vê-se que as perdas, tanto as ativas

como as reativas, são função dos módulos e dos argumentos das tensões em ambos os nós.

No total tem-se um sistema de 4 equações a 12 incógnitas. Como o número de incógnitas tem de ser igual ao número de equações para que o sistema possa ser resolvido analiticamente, vai ter de se especificar 8 das incógnitas e calcular as outras 4.

As 12 variáveis são:

- Quatro potências ativas: P_{G1}, P_{C1}, P_{G2} e P_{C2}.
- Quatro potências reativas: Q_{G1}, Q_{C1}, Q_{G2} e Q_{C2}.
- Duas tensões, com os módulos V_1 e V_2 e os argumentos θ_1 e θ_2.

Veremos agora ver quais as variáveis a especificar e quais a calcular.

Como já foi dito e olhando para as equações (3.22) e (3.24), pode ver-se que as perdas são função dos módulos e dos argumentos das tensões nos nós. Vê-se também que não se consegue calcular os argumentos das tensões, mas apenas a sua diferença. Então não é possível especificar as potências geradas nos dois nós, porque não se conhecem as perdas e sabe-se que estas dependem entre outras, de duas incógnitas que não se conseguem calcular (θ_1 e θ_2).

Então:

1. Estimar as cargas ativas e reativas impostas pelos consumidores (P_{C1}, P_{C2}, Q_{C1} e Q_{C2}). Estas são, ou conhecidas ou facilmente estimáveis.

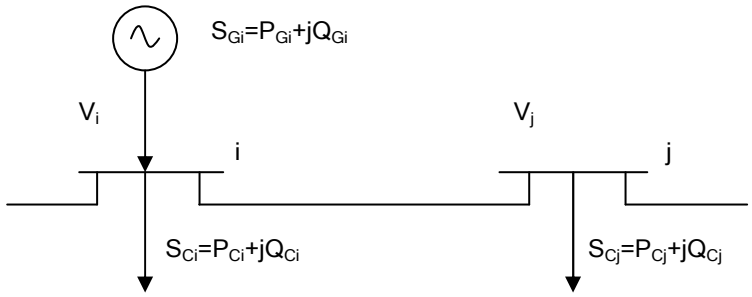
2. Especificar o módulo e o argumento da tensão no barramento 1 (normalmente 1 pu com argumento nulo). Este barramento passa a ser o barramento de referência.

3. Especificar as potências ativa e reativa geradas no barramento 2, que passa a ser um barramento do tipo *PQ*. Neste barramento, uma vez que temos um gerador lá ligado, também poderíamos especificar o módulo da tensão em vez da potência reativa que passaria a ser calculada. Neste caso, o barramento seria do tipo PV (ver tabela 3.1).

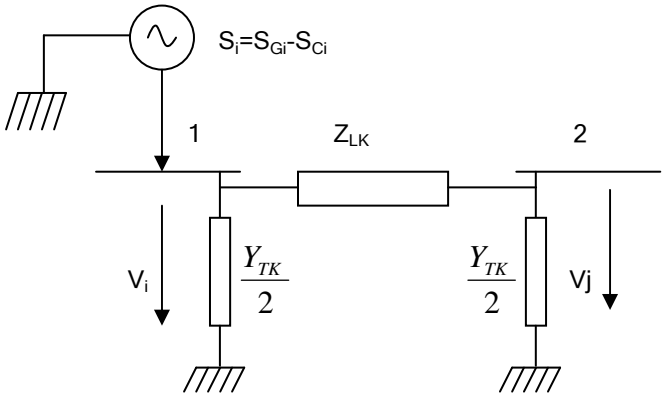
4. Resolver as equações do trânsito de energia para calcular as variáveis restantes, que são agora as nossas incógnitas: o módulo e o argumento da tensão no barramento 2 e as potências ativa e reativa geradas no barramento 1. Devido a este cálculo das potências no barramento 1, este barramento passa também a desempenhar o papel de barramento de balanço, porque é nele que se fecha o balanço energético da rede: energia gerada = energia consumida na carga + energia perdida na linha.

3.2.2. Rede de n barramentos

Na figura 3.10 está representado em a) o esquema unifilar de um nó genérico de uma rede com n nós e em b) o seu modelo equivalente em Π.



a)



b)

Figura 3.10: Rede de n barramentos

a) Esquema unifilar;

b) Esquema equivalente em π

Se na rede de dois barramentos se chegou a duas equações complexas, para o trânsito de energia, nesta rede vai-se chegar a n equações complexas, uma por cada nó da rede.

Seguindo a definição de potência injetada:

$$S_i = P_i + jQ_i = S_{Gi} - S_{Ci} = P_{Gi} - P_{Ci} + j(Q_{Gi} - Q_{Ci}) \tag{3.25}$$

e por analogia com a rede de 2 nós, tem-se para as n equações do trânsito de energia, em notação compacta:

$$P_i - jQ_i = V_i^* \sum_{j=1}^{n} y_{ij} V_j \quad , \text{ i = 1, ..., n} \tag{3.26}$$

$$P_{Gi} - P_{Ci} - j(Q_{Gi} - Q_{Ci}) = V_i^* \sum_{j=1}^{n} y_{ij} V_j \quad , \text{ i = 1, ..., n} \tag{3.27}$$

Substituindo as equações (3.16) e (3.17) em (3.27) e separando a parte real e a parte imaginária, fica, em notação compacta:

$$P_i = P_{Gi} - P_{Ci} = \sum_{j=1}^{n} v_i v_j \left[G_{ij} \cos(\theta_i - \theta_j) + B_{ij} sen(\theta_i - \theta_j) \right], \text{ i = 1, ..., n} \tag{3.28}$$

$$Q_i = Q_{Gi} - Q_{Ci} = \sum_{j=1}^{n} v_i v_j \left[G_{ij} sen(\theta_i - \theta_j) - B_{ij} \cos(\theta_i - \theta_j) \right], \text{ i = 1, ..., n} \tag{3.29}$$

Estas são as 2n equações do trânsito de energia, resultantes da representação das tensões V_i e V_j em coordenadas polares. Elas exprimem o balanço de potência no barramento genérico i.

3.2.3. Solução das equações do trânsito de energia

O primeiro passo na solução das equações do trânsito de energia é o cálculo das tensões (módulo e argumento) nos nós da rede.

3.2.3.1. Cálculo das tensões

Como já foi dito, devido à não linearidade das equações, tem de se utilizar um método iterativo. Existem 3 métodos:

1. Método de Gauss-Seidel.
2. Método de Newton-Raphson.
3. Método de acoplamento.

Em termos de algoritmo matemático, qualquer destes métodos segue a mesma sequência. Inicialmente, é estimado um valor para os módulos e argumentos das tensões em todos os barramentos. De seguida, calculam-se correções para esses valores e somam-se essas correções aos valores estimados. Em cada iteração comparam-se os módulos das tensões, em todos os nós, com os requisitos de precisão especificados. O algoritmo termina quando estes requisitos forem satisfeitos. A solução será tanto mais rigorosa quanto maior for o número de iterações. No entanto, deverá impor-se um limite a este número para evitar um ciclo infinito no caso de o processo ser divergente. Em alternativa poderá usar-se uma heurística que reconheça um padrão de divergência no processo e pare o algoritmo sinalizando esse caso ao utilizador.

Um pouco mais à frente ver-se-á em mais pormenor o método de Gauss-Seidel.

3.2.3.2. Cálculo da potência injetada no nó de referência

Uma vez calculadas as tensões, por um dos métodos mencionados no ponto anterior, estão criadas as condições de calcular a potência injetada no nó de referência/balanço. Para isso utilizar-se a primeira das equações (3.26).

$$P_1 - jQ_1 = V_1^* \sum_{j=1}^{n} y_{ij} V_j \quad , \text{ i = 1} \tag{3.30}$$

Esta equação não entra no processo iterativo descrito no ponto anterior, porque a amplitude e o argumento da tensão, neste barramento, são especificados.

3.2.3.3. Cálculo das potências que transitam nas linhas

Já foram calculadas as tensões nos nós, por um método iterativo (3.3.3.1) e a potência injetada no nó de referência/balanço (3.3.3.2). Agora, vão ser calculadas as potências que transitam nas linhas.

Na figura 3.11 está representada uma linha K, genérica, ligada entre os nós i e j. Consideram-se duas potências. A potência S_{ij}^k, que é positiva no sentido i $\rightarrow$j. A outra potência, S_{ji}^k, é positiva no sentido j $\rightarrow$ i.

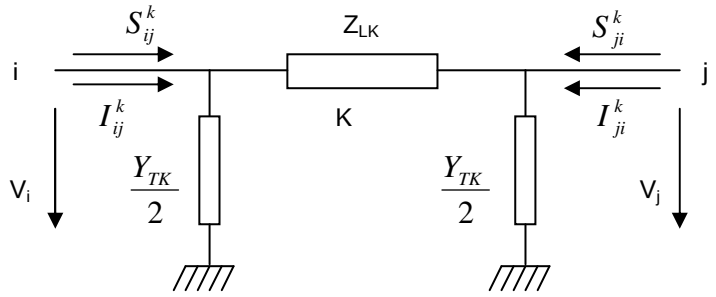


Figura 3.11: Potência transitada numa linha

Então:

$$S_{ij}^k = P_{ij}^k + jQ_{ij}^k$$

$$S_{ij}^k = V_i\left(I_{ij}^k\right)^* = \frac{1}{Z_{L_k}^*}\left(v_i^2 - V_i V_j^*\right) + \frac{Y_{T_k}^*}{2}v_i^2 = \left(\frac{1}{Z_{L_k}^*} + \frac{Y_{T_k}^*}{2}\right)v_i^2 - \frac{1}{Z_{L_k}^*}V_i V_j^* \qquad (3.31)$$

com $$Z_{L_k}^* = R_k + jX_k \qquad (3.32)$$

e $$Y_{T_k} = j\omega C_k \qquad (3.33)$$

onde R_k, X_k e C_k são a resistência, a indutância e a capacitância da linha, respetivamente.

Definem-se as grandezas

$$G_k = \mathrm{Re}\left\{\frac{1}{Z_{L_k}}\right\} = \frac{R_k}{R_k^2 + X_k^2} \qquad (3.34)$$

$$B_k = \mathrm{Im}\left\{\frac{1}{Z_{L_k}}\right\} = -\frac{X_k}{R_k^2 + X_k^2} \qquad (3.35)$$

$$B_k^{'} = \mathrm{Im}\left\{\frac{Y_{T_k}}{2}\right\} = \frac{\omega C_k}{2} \qquad (3.36)$$

e reescreve-se a equação (3.31):

$$S_{ij}^k = \left(G_k - j\left(B_k + B_k^{'}\right)\right)v_i^2 - \left(G_k - jB_k\right)v_i v_j\left(\cos(\theta_i - \theta_j) + jsen(\theta_i - \theta_j)\right) \qquad (3.37)$$

Separando a parte real da imaginária,

$$P_{ij}^{k} = G_k v_i^2 - v_i v_j \left(G_k \cos(\theta_i - \theta_j) + B_k sen(\theta_i - \theta_j)\right) \quad (3.38)$$

$$Q_{ij}^{k} = -\left(B_k + B_k^{'}\right) v_i^2 - v_i v_j \left(G_k \cos(\theta_i - \theta_j) - B_k sen(\theta_i - \theta_j)\right) \quad (3.39)$$

Estas duas equações, (3.38) e (3.39) representam as potências ativa e reativa junto ao nó i (a energia transita do nó i para o nó j).

Por analogia com o nó i e já que a análise para o nó j é igual mas de sentido contrário (potência positiva no sentido j → i), tem-se que:

$$S_{ji}^{k} = P_{ji}^{k} + jQ_{ji}^{k} = V_j \left(I_{ji}^{k}\right)^{*} = \left(\frac{1}{Z_{L_k}^{*}} + \frac{Y_{T_k}^{*}}{2}\right) v_j^2 - \frac{1}{Z_{L_k}^{*}} V_i V_j^{*} \quad (3.40)$$

$$P_{ji}^{k} = G_k v_j^2 - v_i v_j \left(G_k \cos(\theta_j - \theta_i) + B_k sen(\theta_j - \theta_i)\right) \quad (3.41)$$

$$Q_{ji}^{k} = -\left(B_k + B_k^{'}\right) v_j^2 - v_i v_j \left(G_k \cos(\theta_j - \theta_i) - B_k sen(\theta_j - \theta_i)\right) \quad (3.42)$$

E são obtidas as duas equações, (3.41) e (3.42) que representam as potências ativa e reativa junto ao nó j (a energia transita do nó j para o nó i). Somando as equações (3.31) e (3.40) são obtimos as perdas de potência ativa, P_L, e reativa, Q_L, na linha:

$$S_{ij}^{k} + S_{ji}^{k} = P_{ij}^{k} + P_{ji}^{k} + j\left(Q_{ij}^{k} + Q_{ji}^{k}\right) = P_L + jQ_L \quad (3.43)$$

Substituindo as equações (3.38), (3.39), (3.41) e (3.42) em (3.43),

$$P_L = G_k \left(v_i^2 + v_j^2 - 2v_i v_j \cos(\theta_i - \theta_j)\right) \quad (3.44)$$

$$Q_L = -\left(B_k + B_k^{'}\right)\left(v_i^2 + v_j^2\right) + 2B_k v_i v_j \cos(\theta_i - \theta_j) \quad (3.45)$$

3.2.4. O método de Gauss-Seidel

Vai-se ver agora como se resolve o trânsito de energia com o método de Gauss-Seidel. Nos dois casos que a analisar existe um barramento de balanço. No primeiro caso os outros barramentos são do tipo PQ e no segundo são do tipo PV.

3.2.4.1. Barramentos tipo PQ

Das equações (3.26) tira-se que:

$$P_i - jQ_i = V_i^* y_{ii} V_i + V_i^* \sum_{j=1, j\neq i}^{n} y_{ij} V_j \quad , \text{ i = 2, ..., n} \tag{3.46}$$

$$V_i = \frac{1}{y_{ii}} \left(\frac{P_i - jQ_i}{V_i^*} - \sum_{j=1, j\neq i}^{n} y_{ij} V_j \right) \quad , \text{ i = 2, ..., n} \tag{3.47}$$

O i começa em 2 porque i = 1 é o barramento de balanço onde a amplitude e o argumento da tensão são especificados. Tem-se portanto n -1 equações complexas, porque cada tensão tem módulo e argumento. Então, tem-se 2(n -1) incógnitas. Pode então dizer-se que as equações para calcular as tensões nos nós são, em notação compacta:

$$V_i = \frac{1}{y_{ii}} \left(\frac{P_i - jQ_i}{\left(V_i^{k-1}\right)^*} - \sum_{j=1}^{i} y_{ij} V_j^k - \sum_{j=i+1}^{n} y_{ij} V_j^{k-1} \right) \quad , \text{ i = 2, ..., n} \tag{3.48}$$

Para que o processo se inicie tem de se estimar as tensões em todos os barramentos. Dado que num sistema de energia elétrica, a tensão nos diversos barramentos não apresenta diferenças significativas, é uma boa estimativa tomar-se o valor do barramento

de referência para os diversos nós. Esse valor está normalmente compreendido entre 1 a 1,05 p.u. com argumento nulo. O processo termina quando o módulo da diferença das tensões, em todos os barramentos, em duas iterações sucessivas, for inferior a um valor ε.

$$\Delta V_i^k = \left| V_i^k - V_i^{k-1} \right| < \varepsilon \quad , \text{ i = 2, ..., n} \tag{3.49}$$

Tipicamente ε = 10^{-4} p.u.

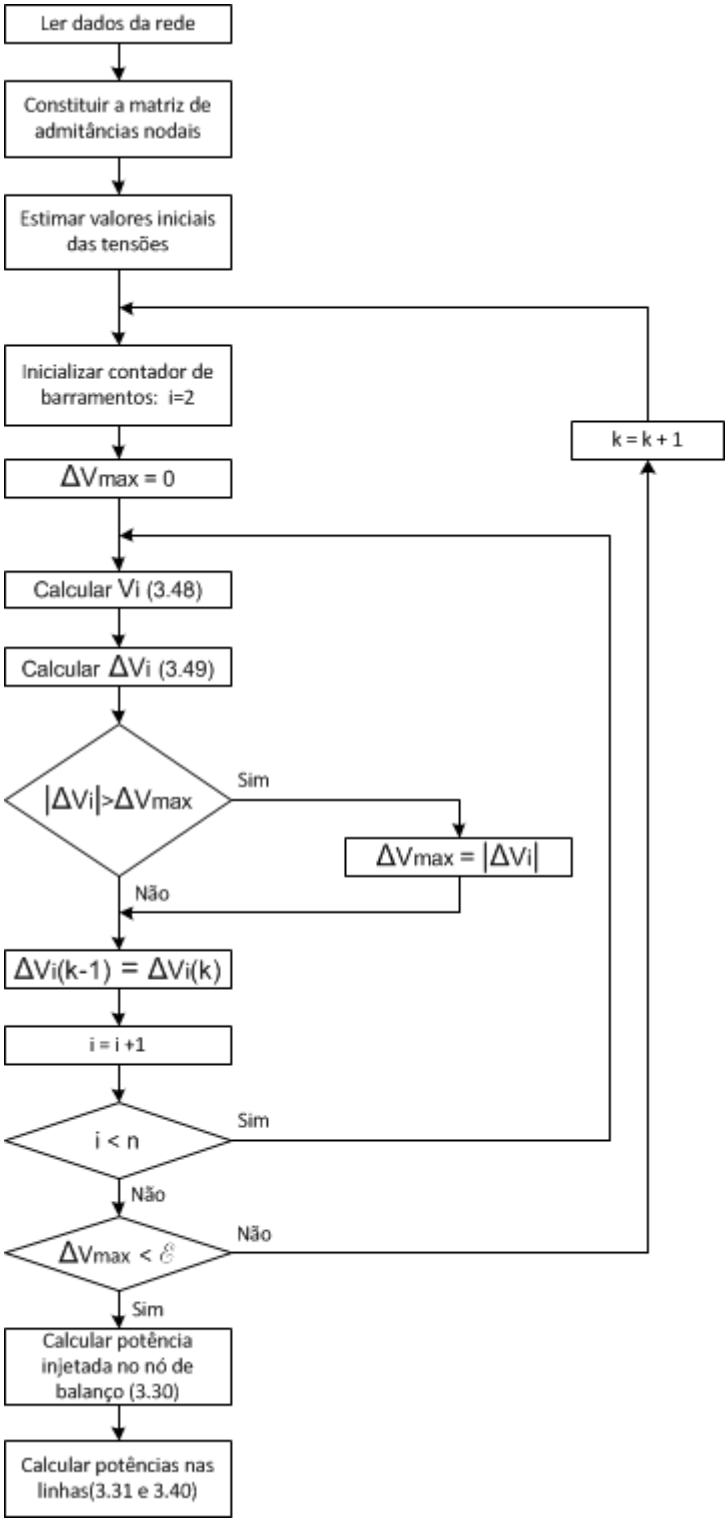


Figura 3.12: Fluxograma para o cálculo do trânsito de energia pelo método de Gauss-Seidel

Comentários ao fluxograma para o cálculo do trânsito de energia pelo método de Gauss-Seidel (Figura 3.12).

Após a leitura dos dados da rede e constituída a matriz de admitâncias nodais, que terá *i - 1* equações, sendo *i* o número de barramentos, estimam-se os valores para as tensões nos nós.

Por cada iteração *k*, a contagem de barramentos é iniciada em 2, porque o barramento de balanço fica de fora do ciclo. Em todos os outros a tensão é calculada. A variável $\Delta V_{\max}$ é inicializada a 0 no início de cada iteração e no fim contem o maior valor calculado nessa iteração (diferença entre todos os barramentos).

No fim de cada iteração o valor de $\Delta V_{\max}$ é testado. Se for maior que ε, inicia-se nova iteração, repetindo-se o ciclo da contagem de barramentos. Se for menor, o processo iterativo termina, sendo calculadas a potência injetada no nó de balanço e as potências transitadas nas linhas. Esta diferença, $\Delta V_{\max}$, é calculada entre todos os barramentos e em duas iterações sucessivas.

3.2.4.2. Barramentos tipo PV

No caso do barramento tipo PV, o que é especificado é o módulo da tensão e não a potência reativa como para os PQ. Então, imediatamente antes do cálculo da tensão, V_i, há que fazer uma alteração no algoritmo, introduzindo aí mais uns passos. Essas alterações ficarão localizadas entre a quinta e a sexta ações do algoritmo da figura 3.12.

Então, começa-se por especificar o módulo da tensão, com a equação (3.50). Este valor é temporário e vai servir para se calcular a potência reativa injetada no nó, através da equação (3.51).

$$V_{it}^{k} = \frac{V_i^{k-1}}{v_i^{k-1}} v_i^{esp} \quad , \text{ i = 2, ..., n} \tag{3.50}$$

$$Q_i^k = -\mathrm{Im}\left[\left(V_{it}^k\right)^* \left(y_{ii} V_{it}^k + \sum_{j=1, j\neq i}^{n} y_{ij} V_j^{k-1} \right) \right] \quad , \text{ i = 2, ..., n} \tag{3.51}$$

Um nó do tipo PV, como se pode ver na tabela 3.1, é um nó só de geração. Portanto há um gerador ligado a esse nó e não há carga. Como é óbvio, esse gerador tem limites mínimo e máximo de potência reativa que pode fornecer. Se o resultado do cálculo da potência reativa for tal que viole um desses limites, fixa-se a potência reativa gerada no valor do limite violado e calcula-se a tensão no barramento com a equação (3.48) (entrada na parte do algoritmo representada na figura 3.13). Significa isto que o nó passou a ser considerado do tipo PQ (especifica-se a potência reativa e calcula-se a tensão), mas um falso PQ, porque não tem carga. Se não houver violação de nenhum dos limites, aceita-se o valor da tensão calculado pela equação (3.50) e recalcula-se esse valor com a equação (3.48) (entrada na parte do algoritmo representada na figura 3.12). Na figura 3.13 está representada a parte do algoritmo modificada/acrescentada para tratar os nós tipo PV.

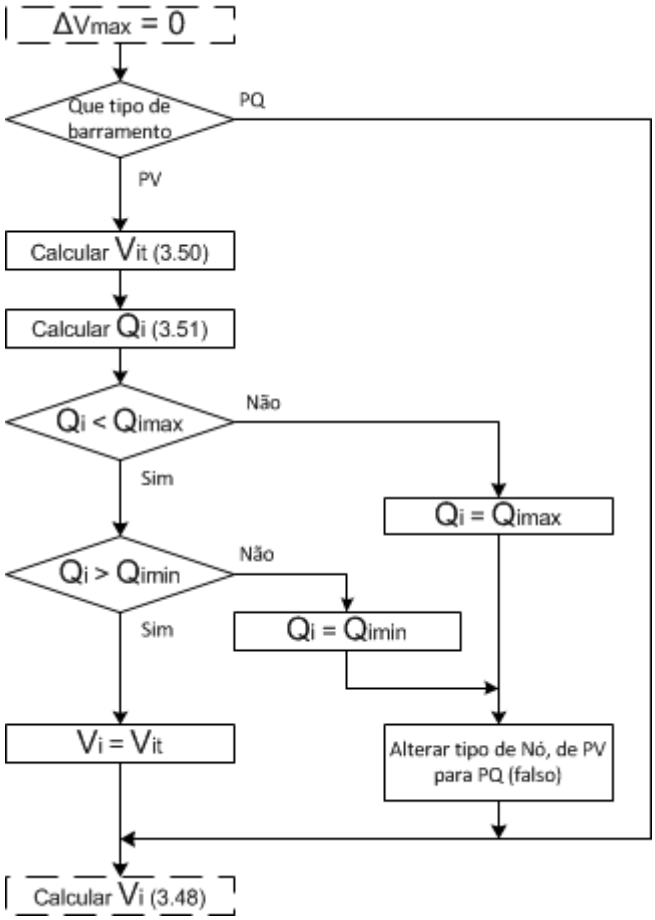


Figura 3.13: Modificações ao fluxograma para tratamento dos nós PV

No início de cada iteração um barramento PV deve ser sempre classificado como tal. A reclassificação dos barramentos deste tipo, ao longo do processo, acontece porque no início os valores calculados para a potência reativa são significativamente diferentes dos finais.

3.3. Implementação dos algoritmos

Os algoritmos a implementar destinam-se a determinar a máxima potência que é possível injetar na rede, em dois casos distintos: injeções não simultâneas e injeções simultâneas.

3.3.1. Injeções não simultâneas

O algoritmo a implementar, destina-se a determinar a máxima potência que é possível injetar em cada nó de uma rede de energia elétrica, sendo essa potência injetada em um nó de cada vez. Os outros nós, os que têm geração, ficam com o valor dessa geração fixo, os que não têm, ficam em 0. Para além do nó em teste, o nó de referência é o único onde a potência muda, para fazer o balanço.

Começa-se por impor restrições físicas na rede elétrica a analisar. Não pode haver linhas em sobrecarga, a potência no nó de referência não pode ser inferior a zero e o módulo das tensões nos barramentos não pode exceder os valores máximos e mínimos determinados. No nó em teste aumenta-se a potência injetada, com o passo, número inteiro a determinar. Por cada iteração é corrido o programa de *Power Flow* e são monitorizadas as linhas, o gerador de referência e o módulo das tensões nos barramentos. O ciclo termina quando alguma das restrições é violada. O valor imediatamente anterior à iteração onde a violação aconteceu é o valor máximo de potência que é possível injetar nesse nó.

Quanto maior for a precisão que se quer ter, mais pequeno é o passo e este pode ser modificado sem necessidade de qualquer outra alteração no algoritmo. Neste trabalho adotou-se um passo de 1 MW. Apresenta-se a seguir (figura 3.14) o fluxograma do algoritmo.

Sempre que se muda de nó são repostos todos os valores que a rede apresentava originalmente, para que a injeção no nó seguinte encontre o valor máximo injetável nesse nó, sem que a rede tenha sido afetada por qualquer outra injeção de potência em qualquer outro nó.

Se se quiser, podemos olhar para este algoritmo como o *PSO* num espaço unidimensional, já que cada partícula só tem uma dimensão (um valor), que é o valor de potência a injetar no nó. A partícula move-se nesse espaço unidimensional, de zero até um valor máximo, com o passo. O único valor que varia de cada vez que a rede é "apresentada" ao *Power Flow* é o valor da potência injetada nesse nó.

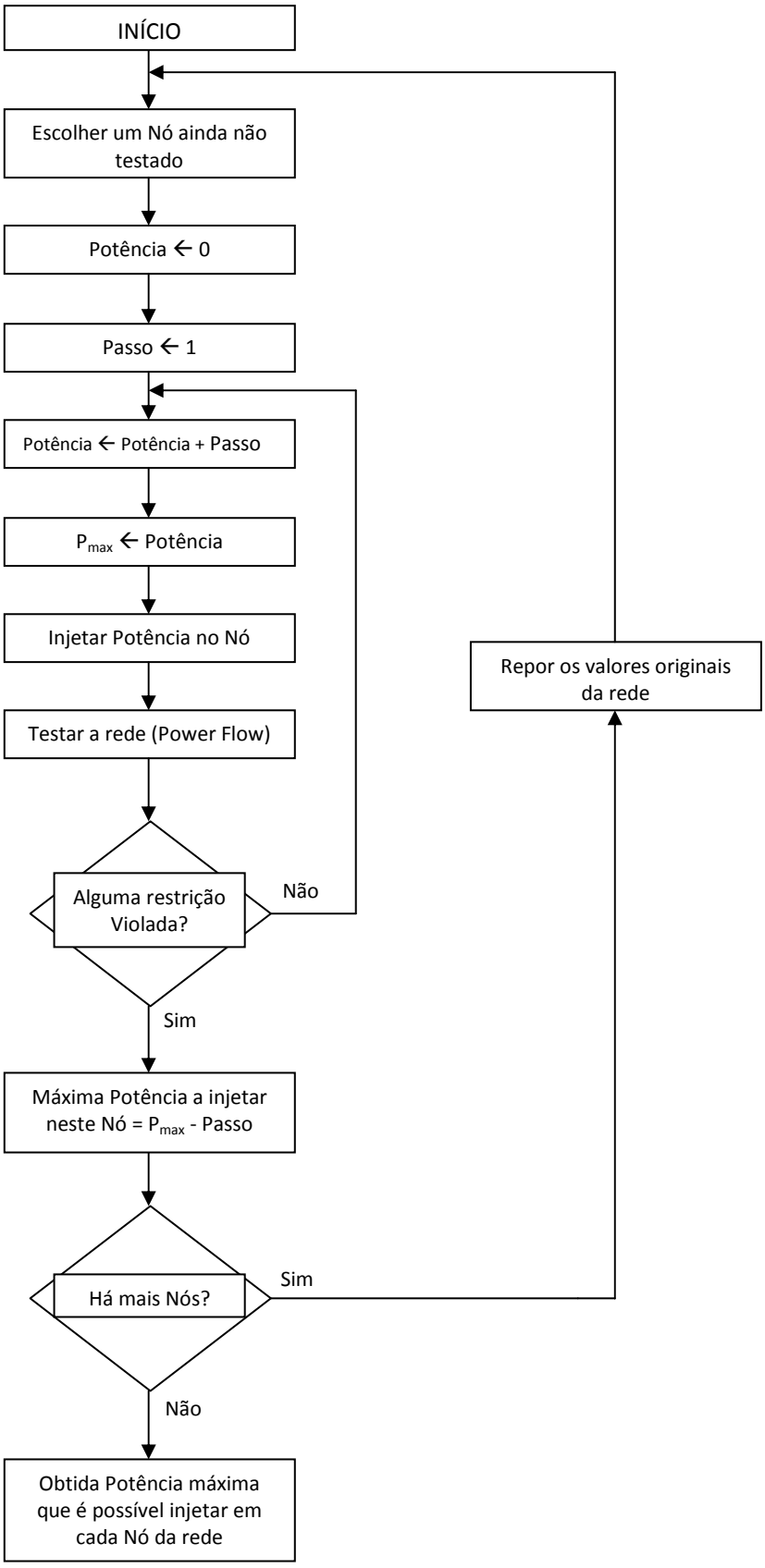


Figura 3.14: Fluxograma do algoritmo para injeções não simultâneas

3.3.2. Injeções simultâneas

Neste caso, o algoritmo *PSO* vai ser adaptado ao problema da máxima injeção nodal simultânea. Para já, pensa-se numa dimensão genérica, d, que representa o número de nós da rede. Neste caso a partícula vai movimentar-se no espaço *d* dimensional. Mais tarde, no subcapítulo 4.3.2. *Máxima injeção nodal simultânea*, será especificada a dimensão para cada uma das redes de teste.

A partícula terá a forma exemplificada no subcapítulo 3.1.2. *Conceito de partícula*. Nesta implementação do algoritmo, não é relevante a forma final das estruturas de dados ou onde estão armazenados os diversos valores. As partículas representam apenas os valores de potência a injetar nos nós, sendo os outros parâmetros referidos pelo seu nome, como P_{best} ou G_{best}, sem a preocupação de se saber onde eles estão fisicamente armazenados.

Para cada nó a potência a injetar varia entre um valor mínimo e um valor máximo. O valor mínimo é zero e o valor máximo, P_{max} é o valor determinado na aplicação do algoritmo da máxima injeção nodal não simultânea. Como no caso anterior, foram impostas restrições físicas à rede a testar. Essas restrições são: não pode haver linhas em sobrecarga, a potência no nó de referência não pode ser inferior a zero e o módulo das tensões nos barramentos não pode exceder os valores máximos e mínimos determinados.

Começa-se por criar o enxame de partículas. Cada partícula é criada com valores aleatórios, para cada nó, entre zero e o seu P_{max}.

No início, o P_{best} (melhor posição encontrada pela partícula até ao momento) de cada partícula é igual à própria partícula, porque até ao momento ela ainda não encontrou nenhuma posição melhor (porque ainda não iniciou o seu movimento). O G_{best} (melhor posição encontrada de todas as partículas, até ao momento) tem a dimensão da partícula e é inicializado com valores aleatórios, entre zero e o menor dos P_{max}. Como se disse na nota da página 24, em 3.1.3, neste algoritmo foi considerado o caso em que a vizinhança da partícula é alargada a todo o espaço de busca, ficando $L_{best} = G_{best}$. Por isso, apenas P_{best} e G_{best} serão considerados para efeitos de atualização da posição da partícula.

Por uma questão de estrutura do algoritmo e simplificação do código, optou-se por atualizar o enxame no primeiro passo da iteração.

Cada partícula representa uma configuração possível de valores de potências nodais para a rede em teste. Para cada uma, resolve-se o trânsito de energia com o método *Newton-Raphson* e vemos se alguma das restrições foi violada. Se houve pelo menos uma restrição violada, os valores das potências nodais da partícula que provocou essa violação são reduzidos e essa partícula não é selecionada para avaliação. Todas as partículas que não tenham provocado violações de nenhuma restrição são selecionadas.

De seguida avalia-se o enxame. Determina-se a melhor partícula da iteração corrente, *k_best_particle* e a melhor partícula de todas as iterações até ao momento, *best_particle*. Para as partículas selecionadas para avaliação, uma partícula será tanto melhor quanto maior for

o somatório Σ dos seus valores de potência nodal, pois é esse o valor que se quer maximizar, de acordo com a função objetivo para o caso da máxima injeção nodal simultânea (2.7). Para cada partícula, se o valor corrente de Σ for maior que o da iteração anterior, significa que a partícula é melhor do que era na iteração anterior e então o P_{best} dessa partícula é atualizado com os valores correntes de potência nodal dessa partícula, como no exemplo da figura 3.6. Se o Σ da *k_best_particle* for maior do que o da *best_particle*, a *k_best_particle* passa a ser a *best_particle* e G_{best} é atualizado com os valores correntes de potência nodal da nova *best_particle*.

O próximo passo é ver se o critério de paragem do algoritmo foi satisfeito. No caso em estudo este critério é uma disjunção de duas condições. O algoritmo pára quando passadas *k* iterações nenhuma partícula consegue obter um Σ melhor do que o corrente ou quando se atinge um número máximo, previamente determinado, de iterações.

De seguida recomeça-se o ciclo. As velocidades e posições das partículas são atualizadas de acordo com as equações (3.1) e (3.2) respetivamente e é de novo resolvido o trânsito de energia para cada uma delas.

Na figura 3.15 apresenta-se um exemplo da progressão da melhor partícula, para o número de iterações *k = 50*. Pode ver-se que, mais ou menos a partir da iteração 38 o valor de Σ estabilizou num patamar de variação mínima, tendo sido constante nas últimas seis iterações. Exemplo com 30 partículas, velocidade mínima de 0 e máxima

de 5, fator de inércia dinâmico respeitando a equação (3.2) e c_1 = c_2 = 1,49618. Pode ver-se no gráfico, pela forma da curva, que a velocidade de deslocamento da melhor partícula, que não é necessariamente a mesma em todas as iterações, embora possa ser, por coincidência, é mais elevada no início do algoritmo e vai diminuindo progressivamente à medida que a partícula se aproxima do patamar.

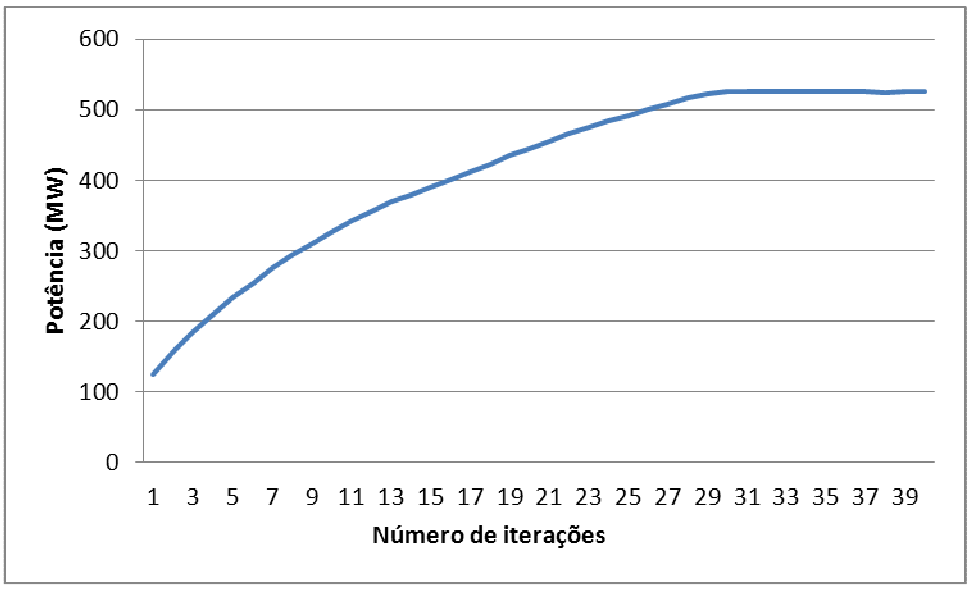


Figura 3.15: Exemplo da progressão da melhor partícula

De acordo com o fluxograma do algoritmo *PSO* para o caso da máxima injeção nodal simultânea, apresentado na figura 3.16, cada iteração é iniciada com a atualização do enxame, uma partícula é indexada, processada pelo método de Newton-Raphson (trânsito de energia), testada (verificação de cumprimento das restrições), avaliada (se é melhor do que era na iteração anterior, se é a melhor desta iteração ou se é a melhor de todas as iterações até ao momento), pas-

sando sozinha por todas as “estações” do algoritmo. No fim, se o critério de paragem não tiver sido satisfeito e se houver mais partículas, são repostos os valores iniciais da rede, é indexada nova partícula e reiniciado o ciclo. Quando já não houver mais partículas, são também repostos os valores iniciais da rede e iniciada nova iteração.

Os valores iniciais da rede são repostos para que cada partícula encontre a rede com a configuração que ela tinha quando se iniciou o ciclo. Estes parâmetros são: as potências ativas e reativas nos nós que têm geração original (ficando todos os outros com zero de potência gerada, tanto ativa como reativa), o módulo e o argumento das tensões em todos os nós.

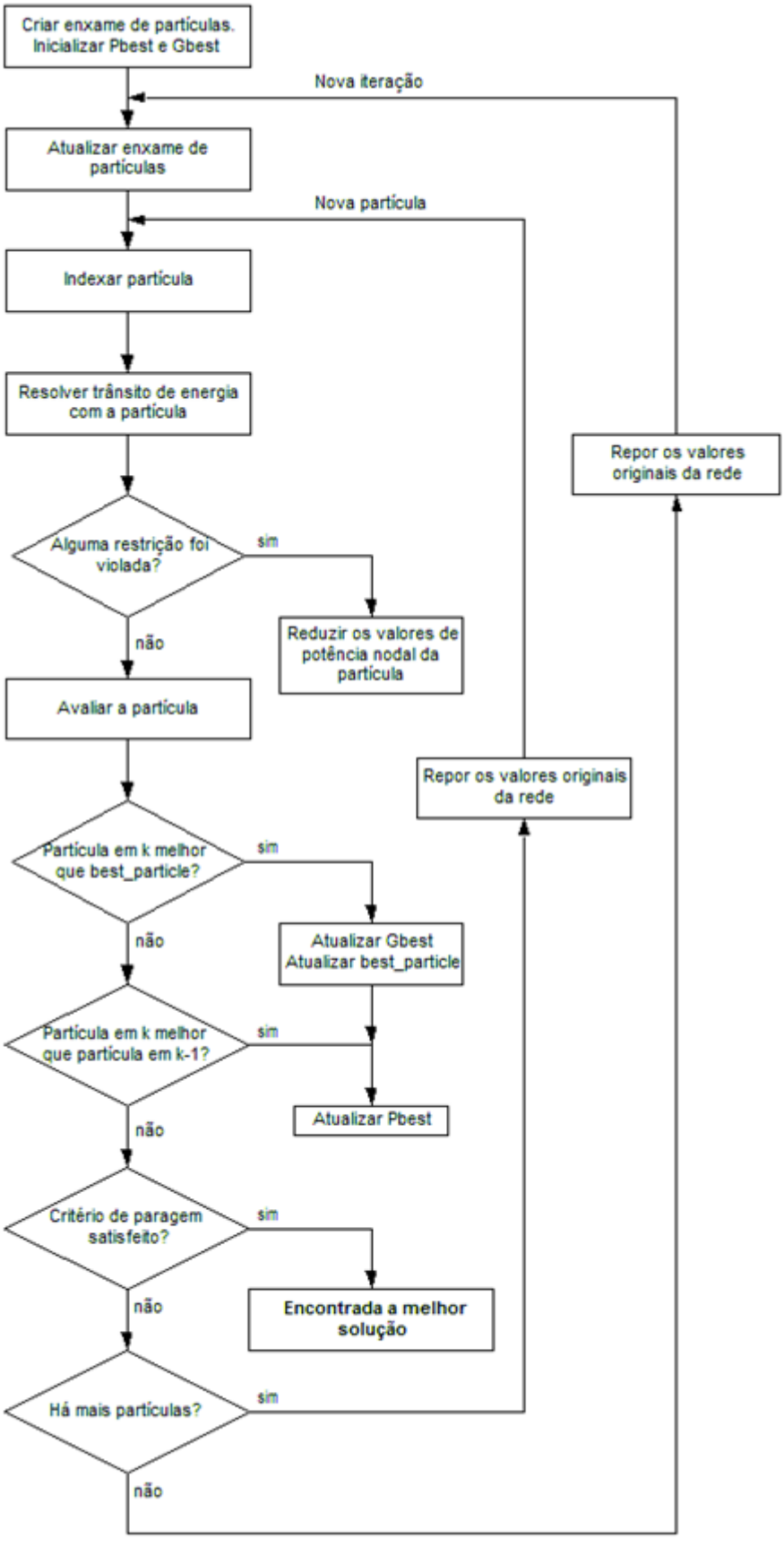


Figura 3.16: - Fluxograma do PSO aplicado à máxima injeção nodal simultanea

Quarto Capítulo - Aplicação

4.1. O ambiente de programação

Como foi visto no capítulo anterior, cada partícula representa uma rede com *n* nós. Cada uma destas redes (partículas) vai ser analisada, resolvendo-se o *trânsito de energia*. Isto é feito pelo programa *PSSE University 33* [15]. Este programa recebe os parâmetros da rede (valores estimados e especificados) e calcula o módulo e o argumento da tensão nos barramentos PQ, o valor do argumento da tensão e a potência reativa nos barramentos PV e as potências ativa e reativa no barramento de balanço. Verifica também se os limites máximos e mínimos das tensões e das potências são respeitados e se o *RATE* das linhas não é ultrapassado.

Por cima deste programa existe uma aplicação de controlo, que gera o enxame de partículas, fornece os parâmetros da rede ao *PSSE University 33*, juntamente com a partícula, dá a ordem de início à função que implementa o *Power Flow*, recebe os resultados (se alguma restrição foi violada ou não), avalia a partícula à luz destes resultados

e faz o restante processamento, iniciando novo ciclo ou parando o algoritmo se estiverem satisfeitos os critérios de paragem. Esta aplicação, que controla o algoritmo *PSO* e o programa *PSSE*, foi escrita na linguagem de programação *Python* [16][17][18][19], versão 27. Escolheu-se esta linguagem, devido à sua versatilidade e porque o programa *PSSE University 33* tem uma API (Application Program Interface) dedicada a esta linguagem, que é o módulo PSSPY. A interface de comunicação com o *PSSE University 33* foi escrita usando as funções deste módulo.

O ambiente de programação utilizado foi o *Eclipse* [20], versão *Kepler*, *Service release* 2, com o módulo Pydev [21]. Foi escolhido pela facilidade de utilização e pelas suas funcionalidades de análise e deteção de erros em tempo real.

4.2. Arquitetura do programa

Antes de ser apresentada a arquitetura propriamente dita, vão ser apresentadas e justificadas algumas opções que foram feitas na escolha dos parâmetros do algoritmo.

Vizinhança: como já foi dito no capítulo 3, optou-se por alargar a vizinhança da partícula a todo o espaço de busca do algoritmo, porque não há interesse em encontrar máximos locais, nem em que as partículas se dispersem em buscas locais, que aumentariam o tempo de computação na procura da melhor solução. É portanto conveniente que a informação da melhor partícula, aquela que está mais perto da solução, seja disponibilizada para todas as outras.

Existirão problemas, que devido à sua especificidade, necessitem de procuras locais, que seja relevante para a sua solução global, a otimização de soluções locais, mas não é esse o caso deste trabalho.

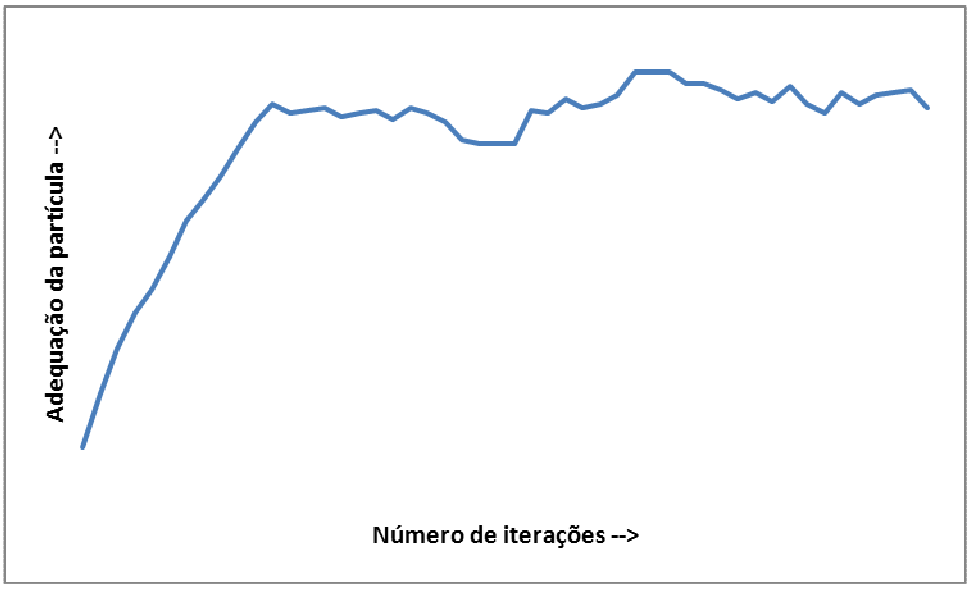
Inércia: depois de vários testes, optou-se por escolher um fator de inércia dinâmico [11] e [12], como já referido em *3.1.3 Parametrização*, sendo maior no início, tem o seu valor mais elevado na primeira iteração e vai gradualmente diminuindo com o número de iterações até atingir o seu menor valor, na última. Deste modo, e como este fator influencia a velocidade da partícula, faz-se com que no início a velocidade seja mais alta, para as partículas poderem "andar mais depressa" e se espalharem mais pelo espaço a explorar e vá diminuindo à medida que estas vão convergindo para o espaço do plano mais perto da melhor solução. Aqui quer-se que a velocidade seja baixa, para que todas as partículas explorem com mais detalhe todo o espaço, porque a melhor solução está perto. Deste modo podem ser evitadas situações em que a partícula se desloca em torno de valores sem encontrar a melhor solução, que pode estar entre eles, mas que a partícula não os vê devido à sua alta velocidade.

Um exemplo real ilustrar esta situação. No caso da máxima injeção nodal, a melhor solução é o maior valor do somatório de todos os valores de potência ativa aplicados a cada nó, sem que viole nenhuma das restrições impostas à rede. Quando uma partícula viola uma restrição os seus valores de potência nodal (ver o *Conceito de partícula* em 3.1.2) são diminuídos de 5%. Na próxima, ou próximas iterações essa partícula vai novamente aumentando esses valores, devido à atualização

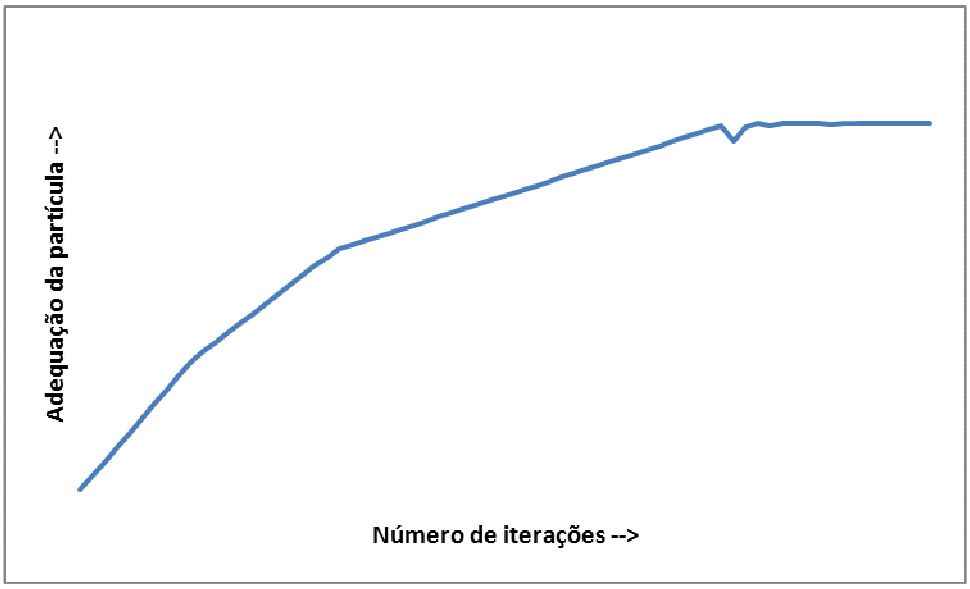
do seu estado, pela equação (3.3), onde à posição da partícula em *t* é somada a velocidade em *t+1*. Se a velocidade for demasiado alta, pode acontecer o que se passa na figura 4.1 a), onde rapidamente a partícula "salta" novamente para um valor onde uma restrição é violada, os seus valores são novamente diminuídos e o ciclo repete-se, podendo a partícula andar, por exemplo a saltar entre os valores 100 e 105 e nunca encontrar o 101, o 102, o 103 ou o 104 onde poderá estar a melhor solução. Se a velocidade for mais baixa (figura 4.1 b)), as partículas podem demorar um pouco mais de tempo a atingir o patamar, mas chegadas aí a procura é mais minuciosa sendo maior a probabilidade de ser encontrada a melhor solução. Fazendo a velocidade variar ao longo do tempo, diminuindo com o número de iterações a eficiência do algoritmo pode ser aumentada.

No caso da velocidade mais alta (figura 4.1 a)), as variações de posição da partícula, no patamar, parecem um pouco aleatórias. Isto deve-se ao fato de duas das componentes da velocidade da partícula serem multiplicadas por dois coeficientes aleatórios, r_1 e r_2, cujos valores pertencem ao intervalo [0,1], (equação (3.1)).

De acordo com a bibliografia consultada e após alguns ensaios para "afinação" do algoritmo, optou-se pelos valores de 1,49618 para c_1 e c_2 [13] e pelos valores de 0,9 e 0,4 para w_{max} e w_{min}, respetivamente [22].



a)



b)

Figura 4.1: Influência da velocidade na procura da melhor solução no algoritmo *PSO*. a) velocidade alta; b) velocidade baixa

4.2.1. Injeções não simultâneas

No caso das injeções não simultâneas, em que se injeta potência ativa num nó da rede de cada vez, deixando os outros com a geração já lá existente, que pode ser igual a zero, não se utilizou o algoritmo *PSO*. Como descrito em 3.2.1, escolhe-se um nó da rede e aumenta-se a potência, com o passo, nesse nó, até uma das restrições impostas ser violada. O valor imediatamente anterior a esse (valor da potência ativa antes do último incremento com o passo) é o valor máximo que podemos injetar nesse nó. Passa-se então ao nó seguinte, até que todos os nós da rede tenham sido testados (fluxograma da figura 3.10).

Quando se faz este teste para o nó de referência/balanço, o que se pretende é injetar potência ativa neste nó, então ele não pode continuar a ser referência, porque nesse não se pode impor um valor de potência. Então tem de se deslocar este nó (referência) para outro nó da rede para que na resolução do trânsito de energia possa ser feito o balanço. A aplicação está dividida em três módulos. Exteriores à aplicação existem mais dois. A arquitetura é a representada na figura 4.2.

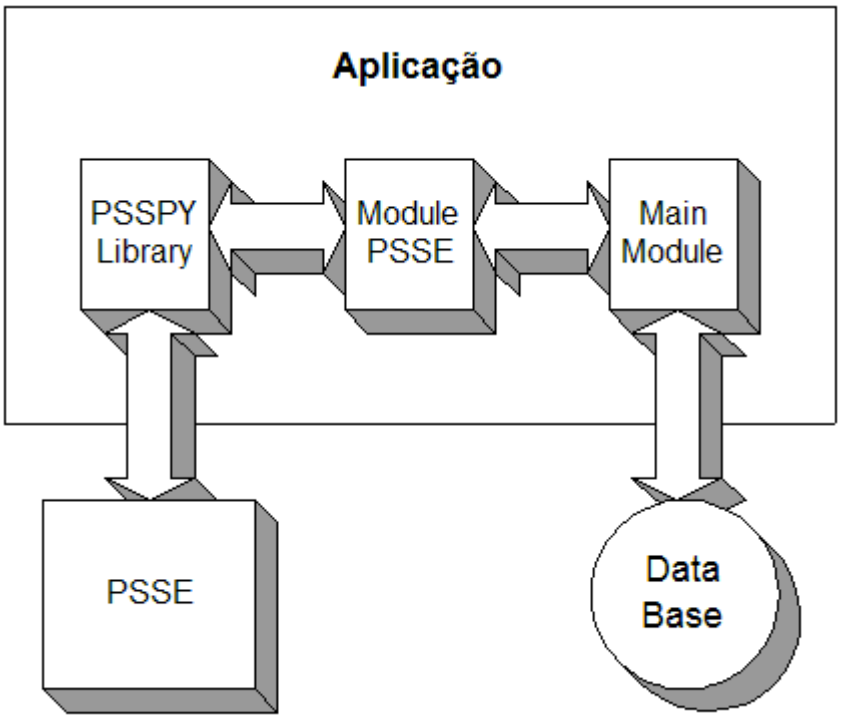


Figura 4.2: Arquitetura da Aplicação para injeções não simultâneas

Os dois módulos exteriores são os seguintes:

1. ***Data Base:*** Uma base de dados de onde são lidos os casos de rede a serem analisados e guardados os resultados, em ficheiros Excel, quando existam gráficos ou em formato de texto quando forem apenas resultados numéricos.
2. ***PSSE University 33:*** O programa que resolve o trânsito de energia da rede a analisar.

Pertencentes à aplicação, existem os seguintes módulos:

1. ***PSSPY Library:*** Este módulo é uma biblioteca que o programa PSSE disponibiliza e que a linguagem *Python* importa para poder usar as funções lá disponibilizadas. São estas funções que escrevem e leem dados nos casos de rede que o programa *PSSE*

analisa. No caso em estudo esta biblioteca é importada pelo módulo *PSSE*, porque é este o módulo que tem a responsabilidade de comunicar com o programa *PSSE*.

2. ***Module PSSE:*** neste módulo estão escritas as funções que, através da biblioteca *PSSPY*, comunicam com o programa *PSSE* para lhe entregar o caso de rede a analisar e os respetivos dados. Esses dados são fornecidos a este módulo pelo *Main Module*, que por sua vez os foi buscar à base de dados. Cada valor de potência ativa para o nó em teste é enviado do *Main Module* para o *Module PSSE* e este, através da biblioteca *PSSPY*, insere este valor no respetivo nó. Depois de resolvido o trânsito de energia para este valor, o módulo *PSSE* lê os resultados do programa *PSSE* e entrega-os ao módulo *Main*, sendo este o responsável por os analisar e tomar as decisões do caminho a seguir pelo algoritmo, de acordo com os resultados da análise.

3. ***Main Module:*** Este é o módulo de controlo. É este módulo que agrega toda a informação e toma as decisões de quando recebe informação de que módulo e quando fornece informação e comandos a que módulo. É este módulo também que recolhe as informações necessárias da base de dados, as distribui e depois de as recolher dos outros módulos, as processar e formatar, as armazena na base de dados.

4.2.2. Injeções simultâneas

No caso das injeções simultâneas, há mais um módulo que é o *Module PSO*.

A aplicação envia e recolhe dados do programa *PSSE University 33*. Esses dados são gerados no *PSO* e os dados recolhidos do *PSSE* são utilizados pelo *PSO* para processar e validar as partículas.

Então, a aplicação está dividida em quatro módulos, como se mostra na figura 4.3. Como no caso das injeções não simultâneas existem mais dois módulos exteriores à aplicação.

Nos módulos comuns aos dois casos, descrever-se-ão apenas as diferenças.

A descrição dos módulos exteriores é igual para este caso.

Pertencentes à aplicação, existem os seguintes módulos:

1. ***PSSPY Library:*** este módulo é igual ao do caso anterior (injeções não simultâneas).
2. ***Module PSSE:*** *n*o caso das injeções simultâneas cada partícula é enviada do *Main Module* para o *Module PSSE* e este, através da biblioteca *PSSPY*, insere os dados da partícula na rede. Depois de resolvido o trânsito de energia para essa partícula, o módulo *PSSE* lê os resultados do programa *PSSE* e entrega-os ao módulo *Main*, sendo este o responsável por os analisar e tomar as decisões do caminho a seguir pelo algoritmo, de acordo com os resultados da análise.

3. ***Main Module:*** A descrição deste módulo é igual à do ponto anterior (injeções não simultâneas).

4. ***Module PSO:*** é neste módulo que estão escritas as funções que implementam o algoritmo *PSO*. Este módulo comunica com o m ódulo de controlo *Main Module*. Recebe deste os parâmetros do algoritmo e devolve-lhe as partículas geradas ou atualizadas, a avaliação das mesmas, a melhor partícula de cada iteração e a melhor partícula, até ao momento, de todas as iterações.

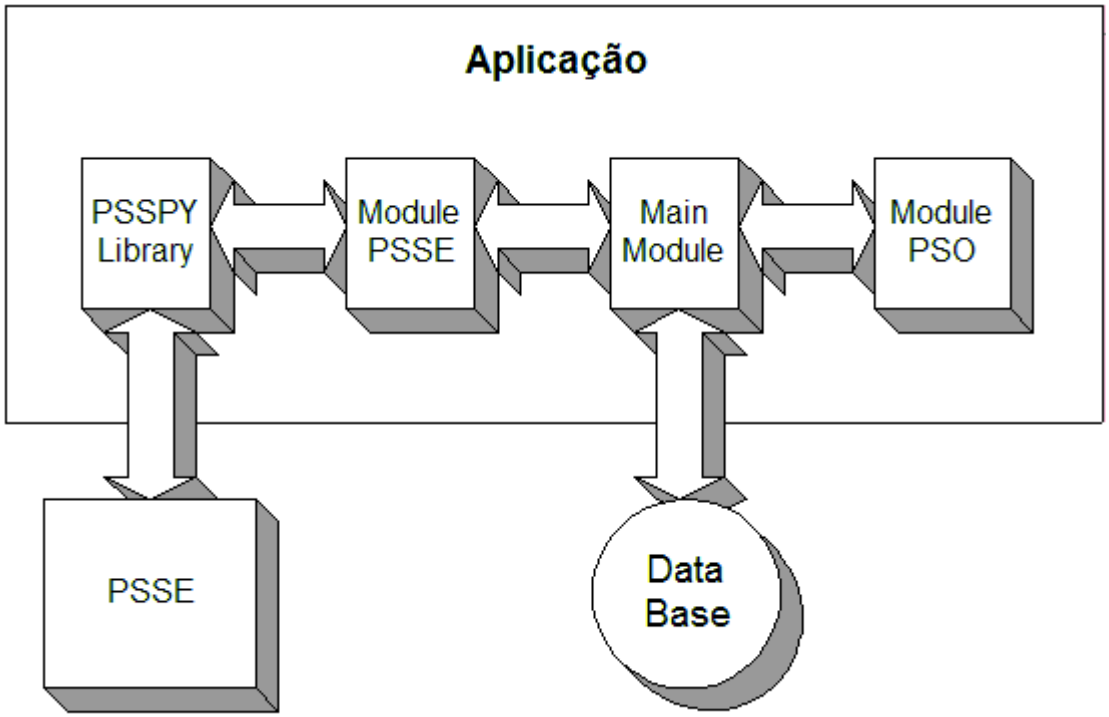


Figura 4.3: Arquitetura da Aplicação para injeções simultâneas

4.3. Aplicação dos algoritmos

Neste subcapítulo vão-se aplicar ambos os algoritmos a duas redes de teste, baseadas nas respetivas redes *IEEE*, uma de seis e outra de catorze barramentos.

4.3.1. Máxima injeção nodal não simultânea

Começa-se pelo caso menos complexo, as injeções não simultâneas.

4.3.1.1. Rede de 6 barramentos

Aplicou-se o algoritmo para as injeções não simultâneas a uma rede de teste de 6 barramentos, baseada na rede de teste *IEEE 6 BUS* [23]. Como já foi dito, neste caso cada nó da rede é tratado separadamente, injetando potência ativa até ser determinado o valor máximo que se pode injetar nesse nó, desde que todos os outros, exceto o de balanço, fiquem com os valores de geração originais. Na figura 4.4, está representado o esquema unifilar da rede de 6 barramentos utilizada, cujos dados se apresentam no Anexo 1.

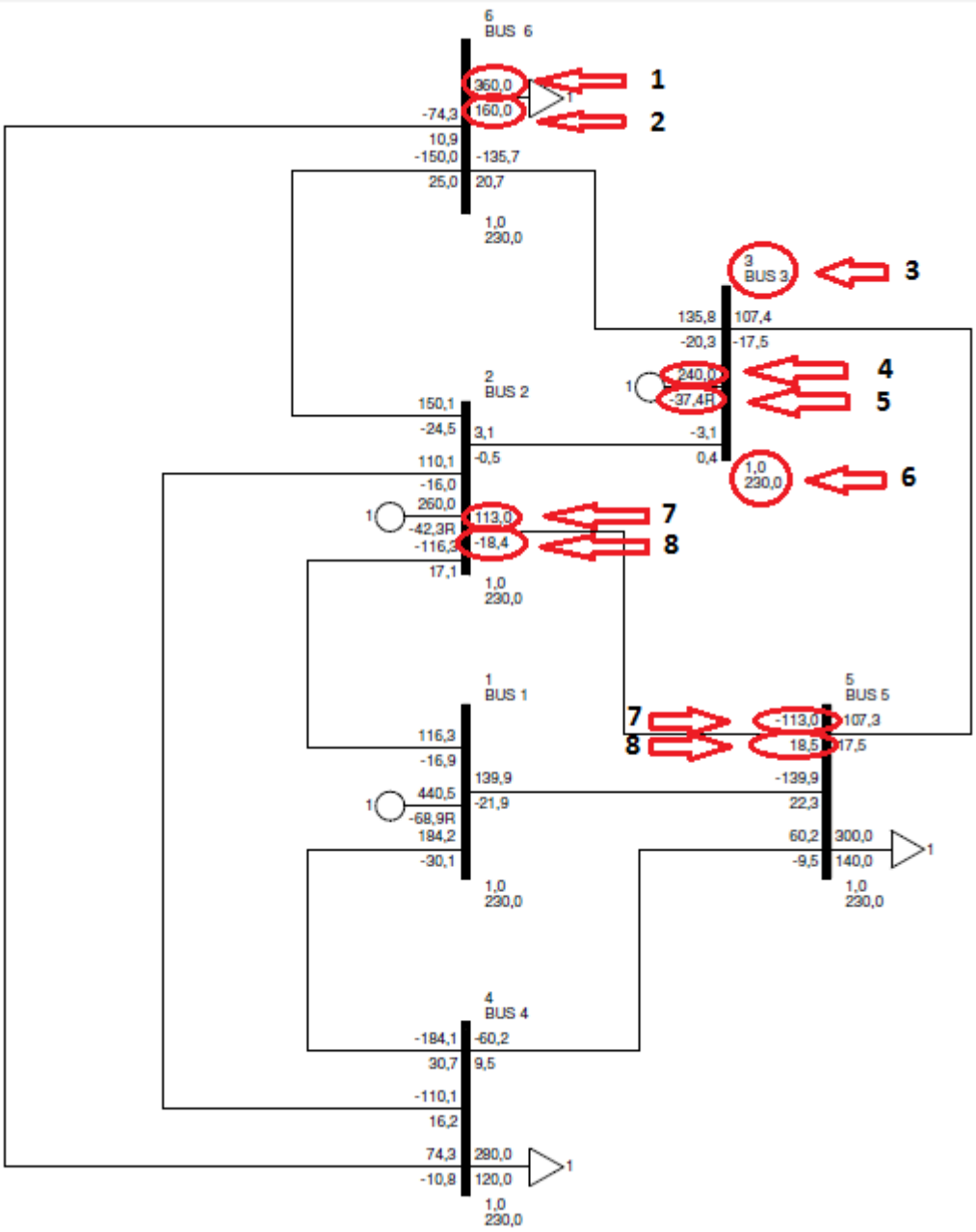


Figura 4.4: Esquema unifilar da rede de teste, de 6 barramentos utilizada, baseada na rede IEEE 6 BUS

Legenda:

1. Potência ativa da carga.
2. Potência reativa da carga.
3. Identificação do barramento.
4. Potência ativa gerada no barramento.
5. Potência reativa gerada no barramento.
6. Tensão do barramento, em pu e em KV.
7. Potência ativa que sai (+) ou entra (-) no barramento.
8. Potência reativa que sai (+) ou entra (-) no barramento.

A seguir (figura 4.5) está representada a mesma rede com a percentagem de ocupação das linhas.

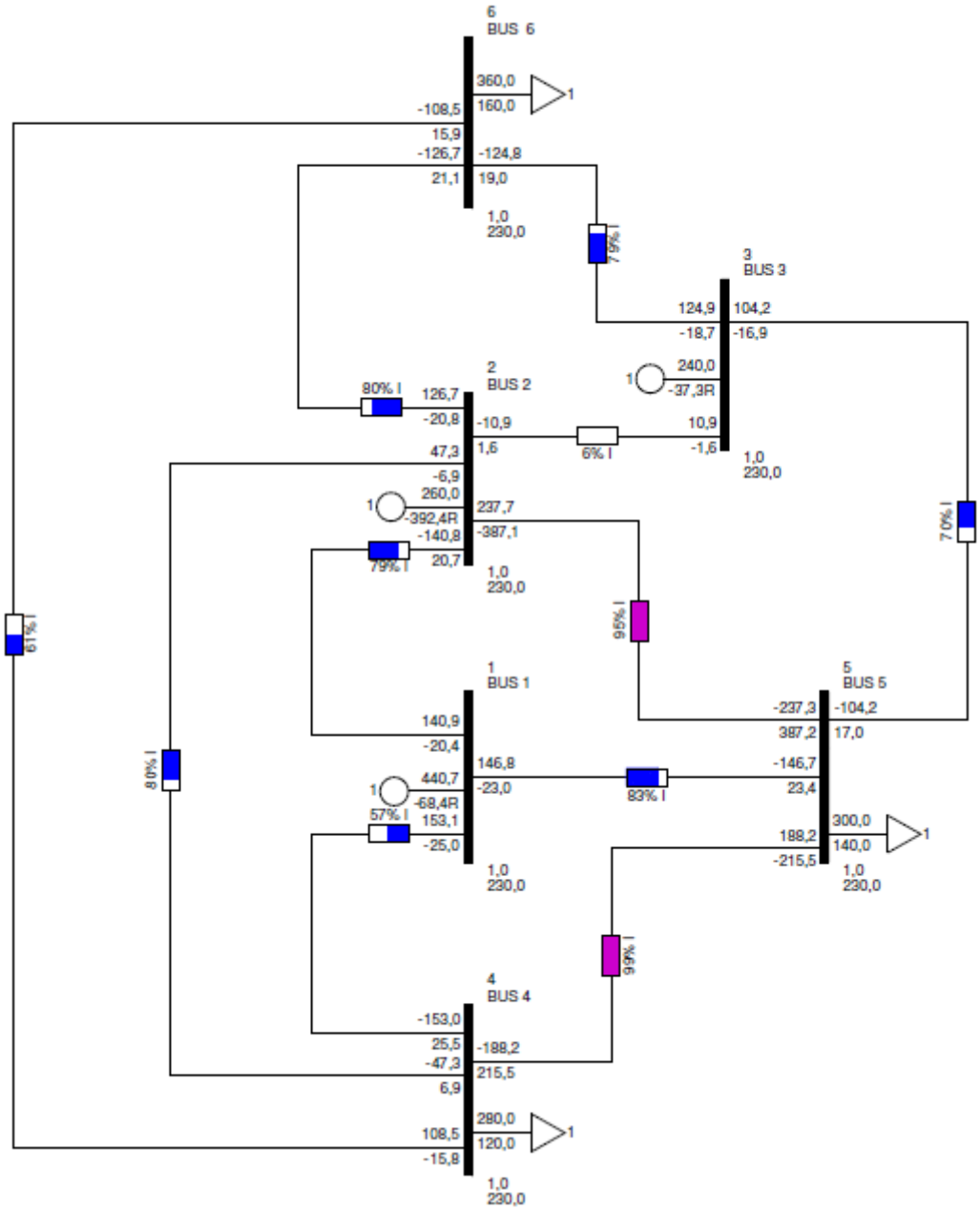


Figura 4.5: Esquema unifilar da rede de teste de 6 barramentos utilizada, baseada na IEEE 6 BUS, com os seus valores originais e mostrando a taxa de ocupação das linhas.

Aplicou-se o algoritmo da figura 3.10 a esta rede (injeção em um nó de cada vez) e obtiveram-se para valores máximos de potência em cada nó, os listados na tabela 4.1.

Tabela 4.1: Valores máximos de potência injetáveis em cada nó

Barramento	1	2	3	4	5	6
P_{max} (MW)	526	270	261	440	30	272

Nos nós onde já havia geração, foi acrescentado outro gerador e foi nesse novo gerador que se foi aumentando a potência ativa injetada, com o passo, até acontecer a violação de uma restrição. Devido a isso, os valores de potência ativa para esses novos geradores nos nós 1, 2, e 3 são 86 MW, 10 MW e 21 MW respetivamente.

Nos dois exemplos apresentados (figuras 4.6 e 4.7), pode-se ver que a injeção no nó 5 está limitada a 30 MW, valor que coloca a linha entre os barramentos 4 e 5 no limite e que a injeção no nó 1 é limitada pela linha entre os barramentos 1 e 2, que atinge 100% da sua ocupação com o valor de 526 MW no nó 1 (86 MW mais do que os já lá gerados). O valor da potência ativa total que esta rede consegue absorver, qualquer que seja o nó onde se injete, é de 940,7 MW.

Com esta análise da máxima injeção nodal não simultânea podemos também identificar pontos da rede onde seja vantajoso reforçá-la para acomodar nova injeção de energia.

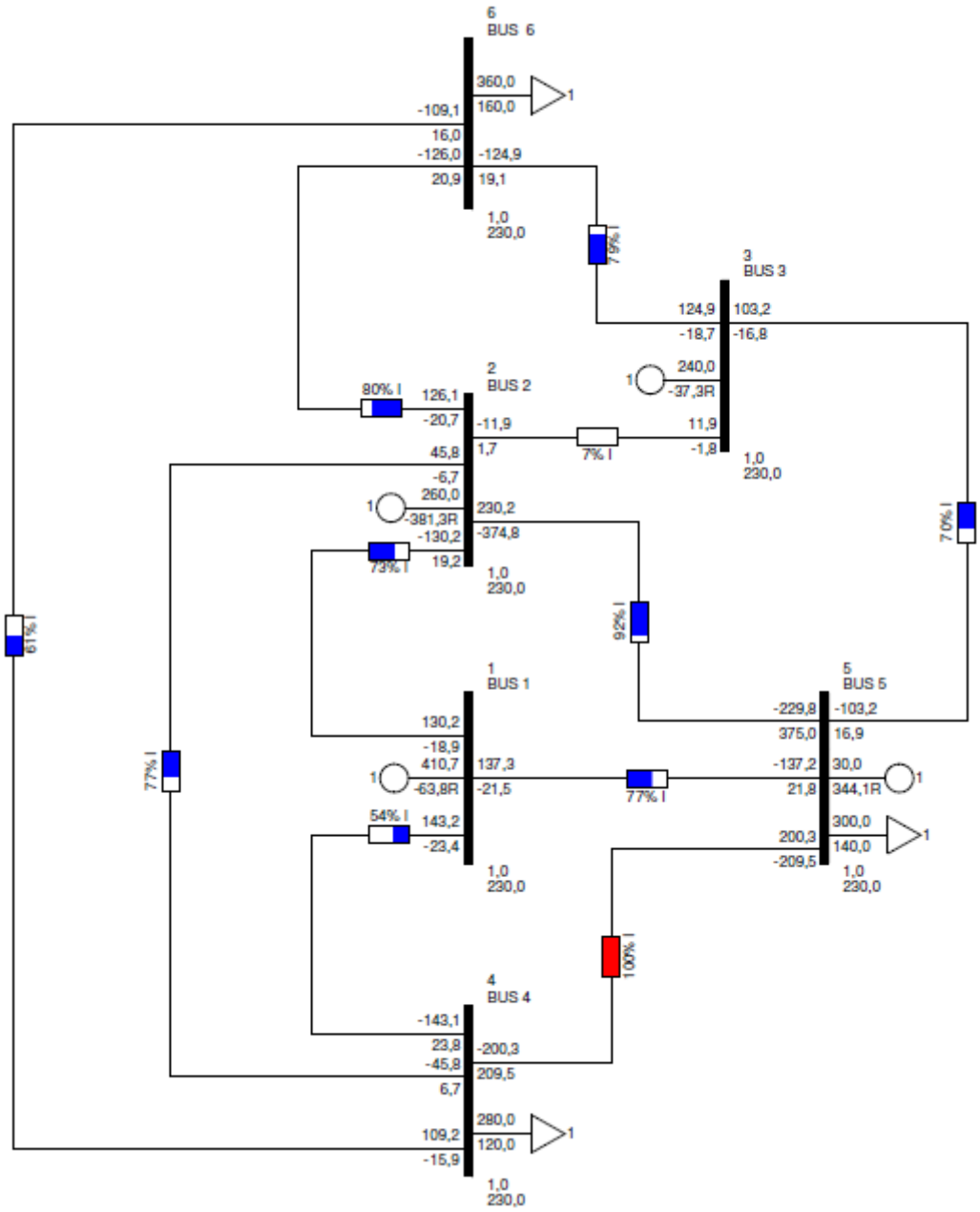


Figura 4.6: Esquema unifilar da rede de teste de 6 barramentos utilizada, com uma injeção de 30 MW no nó 5, mostrando a taxa de ocupação das linhas.

Como se pode observar nas figuras 4.6 e 4.7, embora a potência ativa total absorvida pela rede seja a mesma (940,7 MW, para alimentar as cargas), as taxas de ocupação das linhas são diferentes.

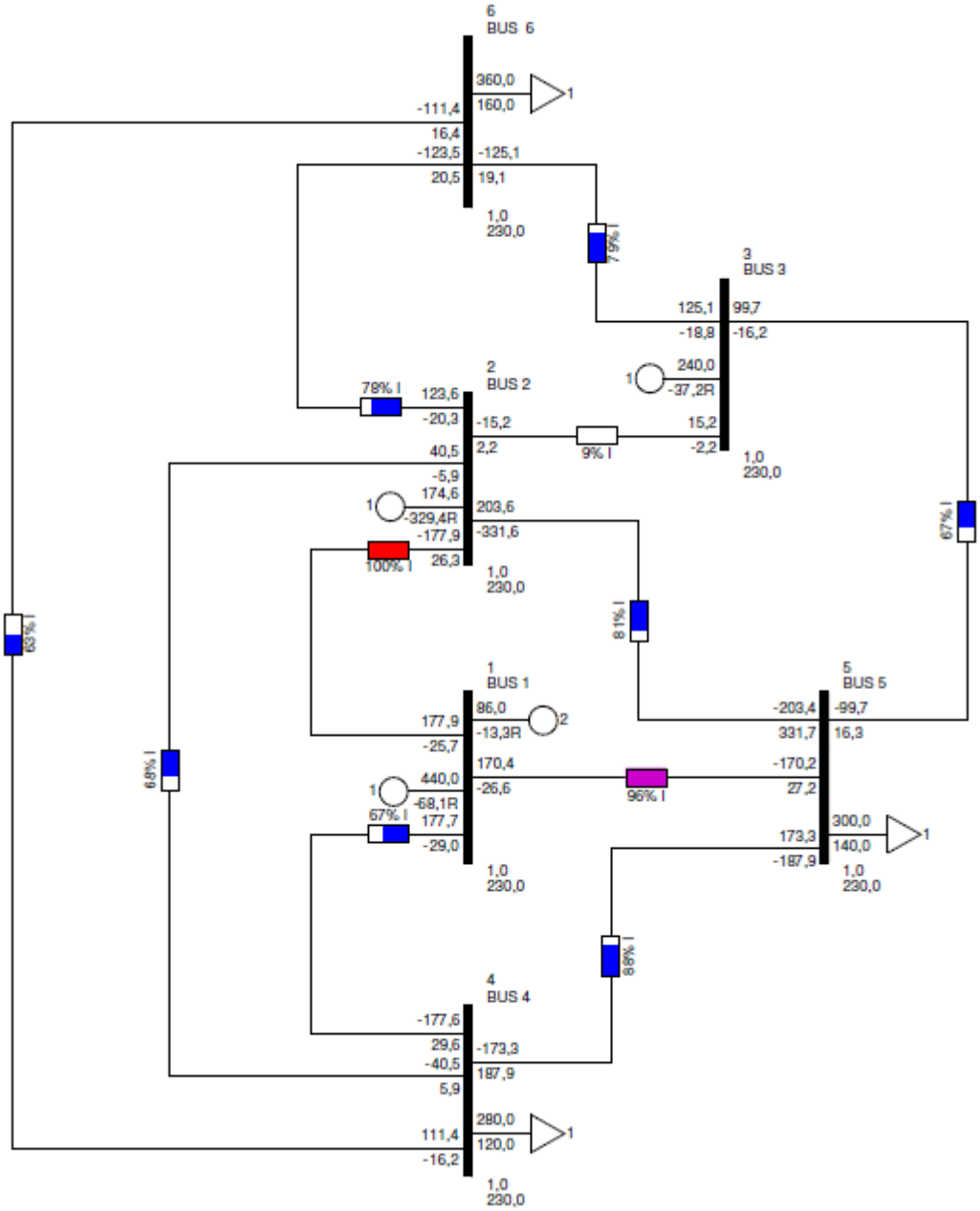


Figura 4.7: Esquema unifilar da rede de teste de 6 barramentos utilizada, com uma injeção de 526 MW no nó 3, mostrando a taxa de ocupação das linhas.

4.3.1.2. Rede de 14 barramentos

Agora vai ser aplicado o mesmo algoritmo, figura 3.10, a uma rede de teste de 14 barramentos, baseada na rede *IEEE 14 BUS* [24]. Na

figura 4.8, está representado o esquema unifilar da rede de 14 barramentos utilizada, cujos dados se apresentam no Anexo 1.

À semelhança do que foi feito para o caso da rede de 6 barramentos, aplicou-se o algoritmo da figura 3.10 (injeção em um nó de cada vez) a esta rede e obtiveram-se para valores máximos de potência em cada nó, os listados na tabela 4.2.

Tabela 4.2: Valores máximos de potência injetáveis em cada nó

Barramento	P_{max} (MW)
1	1051
2	825
3	671
4	311
5	571
6	1321
7	178
8	775
9	186
10	571
11	571
12	452
13	237
14	335

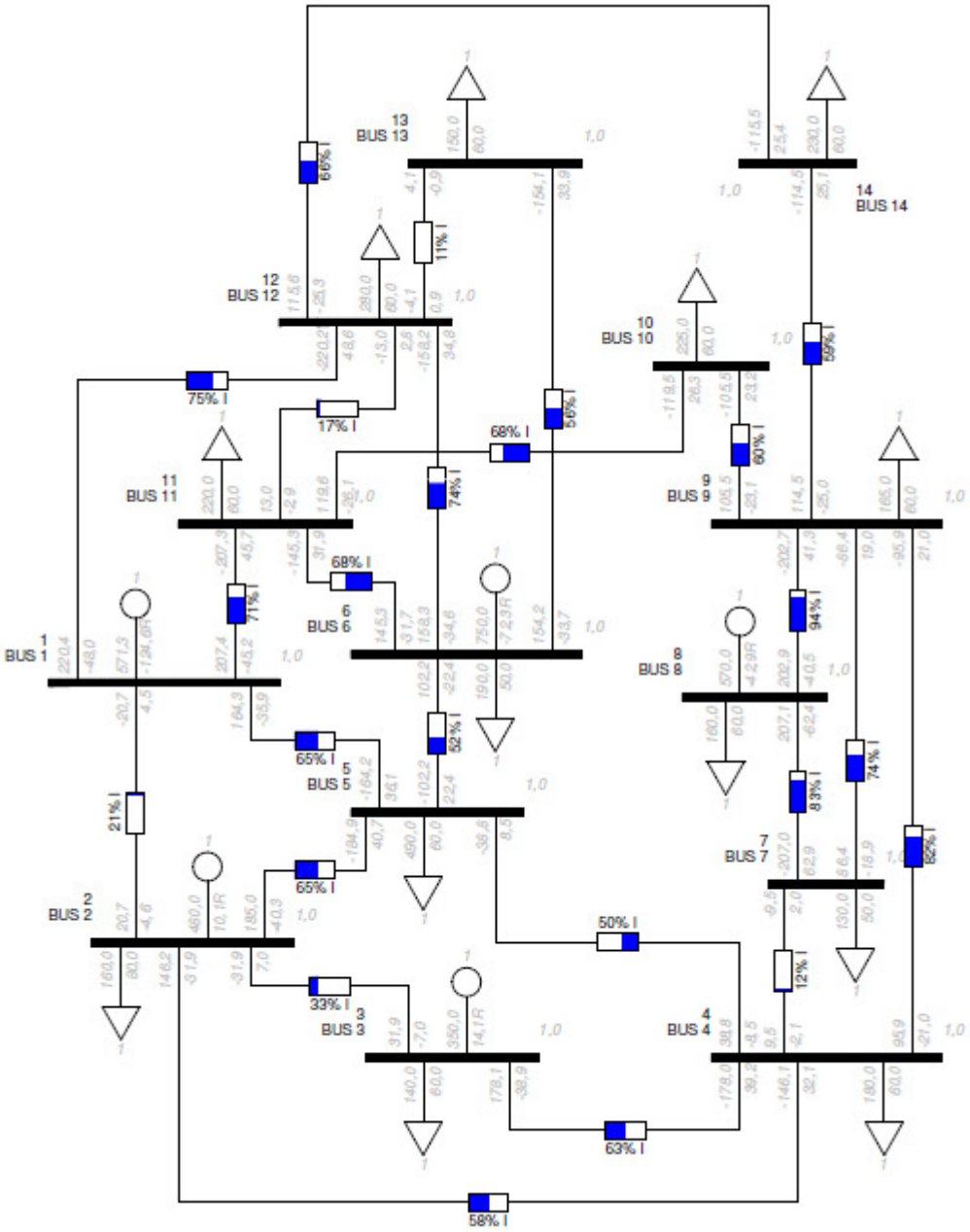


Figura 4.8: Esquema unifilar da rede de teste de 14 barramentos utilizada, baseada na rede IEEE 14 BUS, mostrando a taxa de ocupação das linhas

Também aqui, como na rede de 6 barramentos, nos nós onde já havia geração, foi acrescentado outro gerador e foi nesse novo gerador que se foi aumentando a potência ativa injetada, com o passo, até

acontecer a violação de uma restrição. Devido a isso, os valores de potência ativa para esses novos geradores nos nós 1, 2, 3, 6 e 8 são 480, 345, 321, 571 e 205 MW respetivamente.

Como se vê na tabela 4.2, existem nós onde se pode injetar mais de 1000 MW de potência ativa e outros onde não se chega a 500. Os exemplos mais ilustrativos desta situação são os barramentos 6, com 1321 MW, o maior valor e o 13 com 237, o menor.

Na figura 4.9 é um exemplo do estado da rede quando se injetam 237 MW no nó 13, ficando todos os outros com a geração original. Como se pode ver na figura, esta injeção provoca uma taxa de ocupação de 100% na linha entre os barramentos 12 e 13. Um valor de 238 MW injetado nesse nó, já faria essa linha entrar em sobrecarga, o que não é permitido.

Por analogia com a rede de 6 barramentos, injeções em nós diferentes provocam taxas de ocupação diferentes nas linhas. A potência ativa absorvida pela rede será a necessária para alimentar as cargas instaladas.

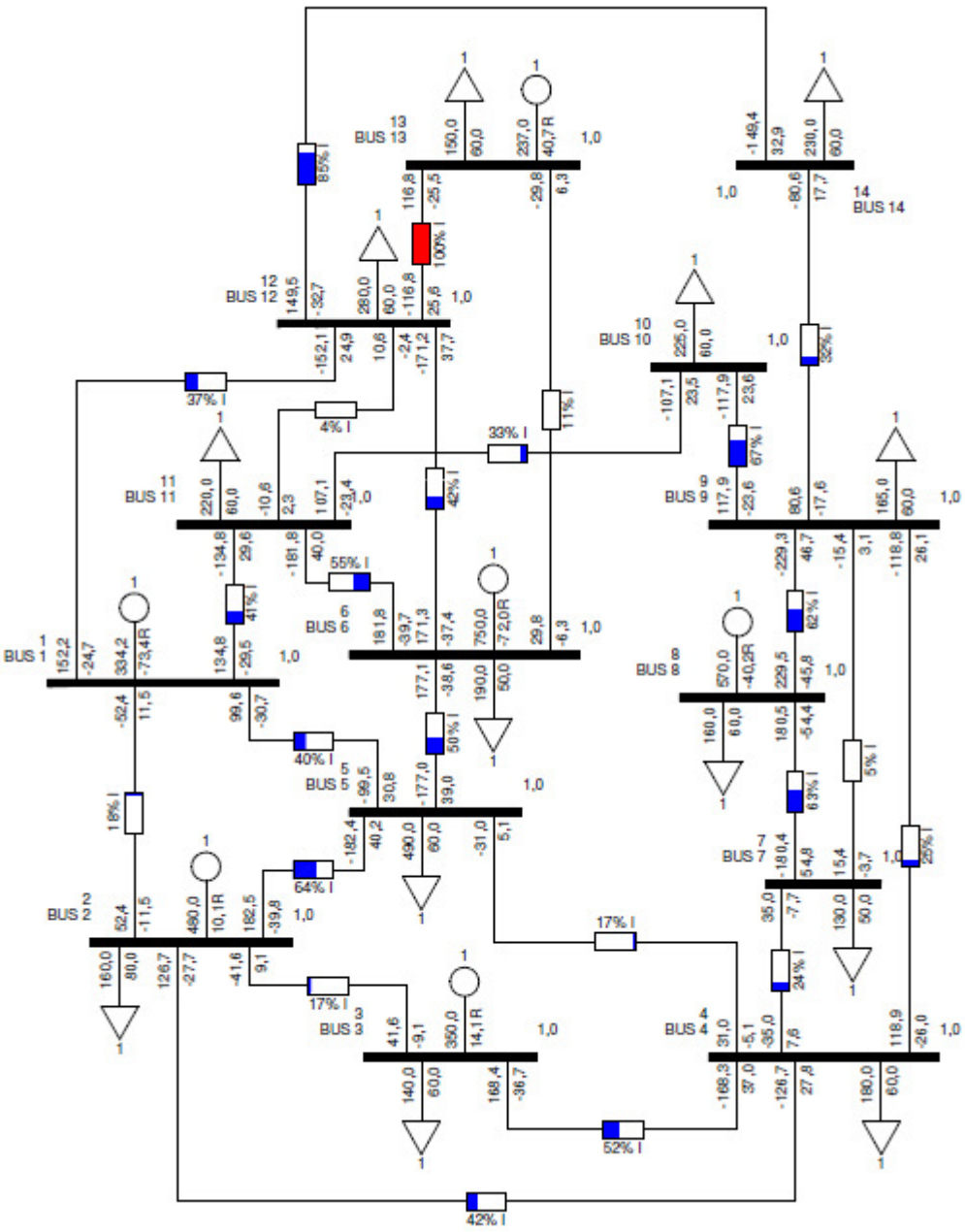


Figura 4.9: Esquema unifilar da rede de teste de 14 barramentos utilizada, com injeção de 237 MW no nó 13, mostrando a taxa de ocupação das linhas

4.3.1.3. Análise dos resultados

Olhando para os resultados obtidos tanto na rede de 6 barramentos como na de 14, vê-se que há linhas com taxas de ocupação baixas e outras com taxas mais elevadas. Também se vê que alterando o nó onde se injeta potência se alteram as taxas de ocupação das linhas. Isso pode ser visto facilmente comparando as figuras 4.6 e 4.7. Então, se alterando o nó da rede onde se injeta a potência ativa se consegue alterar as taxas de ocupação das linhas, pode escolher-se uma combinação de nós e de valores de potência ativa que permitam, em caso de necessidade, injetar mais energia na rede, mas evitando sobrecargas ou mesmo taxas de 100% de qualquer linha.

A essa combinação de valores, que no algoritmo é modulada pela partícula, será chamada a melhor solução. É da resolução desse problema que se vai tratar no ponto seguinte, aplicando o algoritmo *PSO* a estas duas redes (6 e 14 barramentos).

4.3.2. Máxima injeção nodal simultânea

O algoritmo *PSO* vai ser agora aplicado ao problema da máxima injeção nodal simultânea. Como se disse no último ponto, o objetivo é encontrar-se a combinação de valores nodais, de potência ativa que maximizem o total de energia que a rede pode absorver. No final, uma partícula terá esses valores e será essa a melhor solução do trânsito de energia da rede em estudo, encontrada por este algoritmo. Este al-

goritmo, o utilizado neste estudo foi definido para trabalhar com valores discretos, sendo a sua resolução de 1 MW.

Pode também acontecer que existam várias partículas, várias combinações de valores de potência ativa, que maximizem a potência total, o somatório dos valores em todos os nós seja igual, mas que esses valores sejam diferentes em cada partícula. Esta questão será falada novamente nos pontos seguintes quando o algoritmo da injeção nodal simultânea for aplicado às redes de 6 e 14 barramentos.

4.3.2.1. Rede de 6 barramentos

Para este caso vai ser utilizada a mesma rede de 6 nós (figura 4.5), que foi utilizada para o caso das injeções não simultâneas. A esta rede vai ser aplicado o algoritmo cujo fluxograma está representado na figura 3.17.

Recordando o que se disse no ponto 3.3.2, para cada nó a potência a injetar varia entre um valor mínimo e um valor máximo. O valor mínimo é zero e o valor máximo, P_{max} é o valor determinado na aplicação a esta rede do algoritmo da máxima injeção nodal não simultânea (tabela 4.1). Após alguns ensaios, foi encontrado como um bom valor para a velocidade o valor 10. Todos os outros parâmetros ficaram conforme exposto em 4.2.

Na figura 4.10 apresenta-se em modo gráfico o resultado da aplicação do algoritmo à rede de 6 barramentos. Estes resultados foram obtidos com uma população de 30 partículas e com a velocidade da

partícula limitada entre um valor mínimo $v_{min} = 0$ e um valor máximo $v_{max} = 10$. O critério de paragem é quando o algoritmo deteta o mesmo valor máximo dez vezes seguidas e não consegue em nenhuma dessas iterações obter nenhum valor superior a esse. Isto significa que o valor imediatamente a seguir, dentro da resolução definida (1 MW) já vai violar pelo menos uma das restrições impostas à rede. A linha azul representa o andamento da melhor partícula de cada iteração, que pode ser diferente de iteração para iteração, embora possa repetir-se por várias ou até mesmo por todas as iterações. A linha a verde representa a média de todas as partículas em cada iteração, que será sempre de valor inferior ao da melhor partícula, convergindo para um valor próximo desse com o aumento do número de iterações.

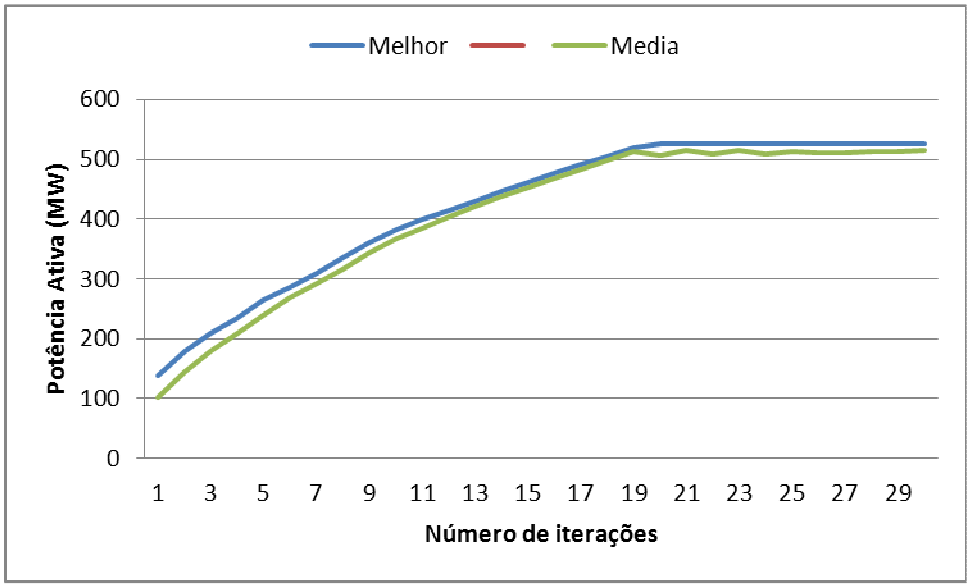


Figura 4.10: Progressão da melhor partícula e da média de todas as partículas com o número de iterações

As oscilações que a curva da média apresenta a partir da iteração 19, devem-se ao fato de a partir dessa iteração começar a haver um número substancial de partículas que provocam violação de restrições. A atuação do algoritmo, neste caso, é diminuir os valores nodais dessas partículas em 5%, o que baixa o valor da média. Este efeito nota-se tanto mais quanto mais partículas houver nessas condições. A partir da iteração 25, devido à redução de velocidade e à aprendizagem das partículas, este efeito vai-se reduzindo e a média das partículas vai convergindo para o valor da melhor partícula.

Na tabela 4.3 estão os valores que deram origem ao gráfico.

Tabela 4.3: Progressão da melhor partícula e da média de todas as partículas, com o número de iterações

Iteração	0	1	2	3	4	5	6	7	8	9
Média	102	145	178	208	239	268	293	317	343	367
Melhor	139	180	208	235	264	287	209	336	360	382
Iteração	10	11	12	13	14	15	16	17	18	19
Média	385	403	421	438	454	469	483	498	513	507
Melhor	399	415	430	446	461	476	491	505	519	525
Iteração	20	21	22	23	24	25	26	27	28	29
Média	514	508	514	508	512	511	511	512	512	515
Melhor	526	526	526	526	526	526	526	526	526	526

Como se pode ver na figura 4.11, e como já foi dito, nos barramentos onde já havia geração, foi criado um novo gerador, gerador 2 e foi nesse que foi feita a injeção de potência ativa. Nos barramentos em que não havia geração, foi criado um gerador, gerador 1 para a potência poder ser injetada.

No exemplo da figura 4.11, foram injetados na rede os valores correspondentes à partícula que o algoritmo selecionou como a melhor solução. A da tabela 4.4.

Tabela 4.4: A melhor partícula encontrada pelo algoritmo *PSO* quando aplicado à rede da figura 4.11

Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Total
86	10	21	208	30	171	526

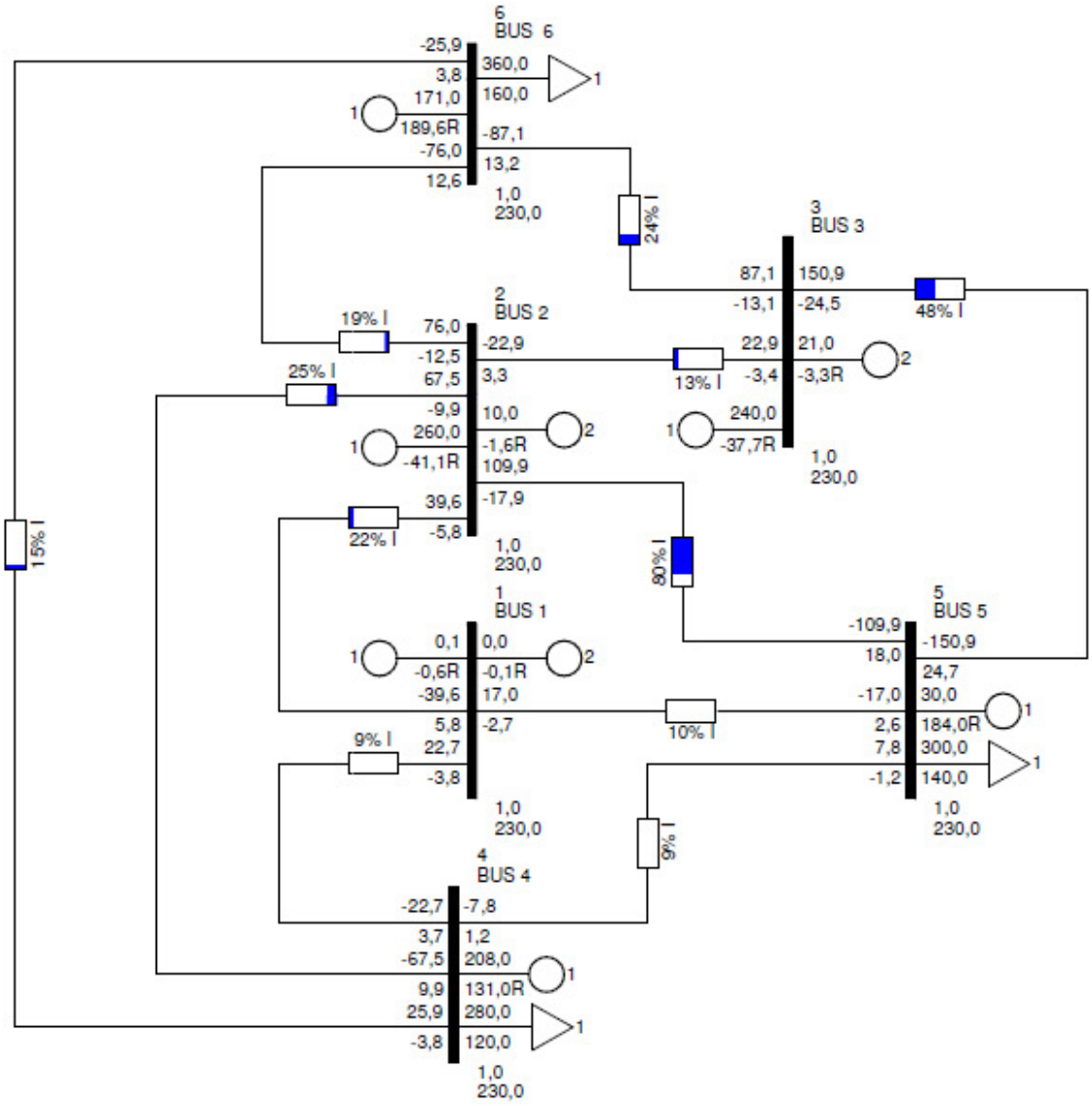


Figura 4.11: Rede de 6 barramentos, quando aplicados os valores de potência nodal da melhor partícula, mostrando a taxa de ocupação das linhas

De notar, na figura 4.11, que o valor de potência ativa no barramento de referência/balanço, barramento 1, é muito perto de 0, considerando a resolução definida (1 MW). Isto quer dizer que esta partícula tem uma configuração que corresponde ao valor máximo, de potência ativa, que podemos injetar nesta rede.

Pode haver mais do que uma partícula que corresponda ao valor máximo de potência ativa que é possível injetar nesta rede. Veja-se o exemplo da tabela 4.5. A posição mais à direita é o somatório dos valores injetados nos nós.

Os valores apresentados nas partículas da tabela 4.5 nos nós 1, 2 e 3 são os valores de potência ativa adicionais, porque esta rede já tinha geração nos barramentos 1, 2 e 3. Em cada uma destas três partículas o somatório dos valores nodais é de 526 MW, que é igual ao valor que se obteve no caso das injeções não simultâneas, quando se injetava apenas no nó 1. É o valor de potência que, depois de resolvido o trânsito de energia, leva o nó de balanço para um valor o mais perto possível de zero (com uma aproximação definida como aceitável), mas ainda positivo.

Quando isto acontece, o gestor da rede pode tomar decisões com base em informação adicional, por exemplo, as perdas de energia nas linhas, que cada uma das combinações apresenta. No exemplo da tabela 4.5, embora as três partículas apresentem o mesmo valor de somatório das potências nodais (526 MW), as perdas são de 829,11 KW em a), 838,64 KW em b) e 833,15 KW em c).

Tabela 4.5: Três partículas diferentes, mas com igual valor de somatório das potências ativas injetadas nos nós

Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Total
86	10	21	209	30	170	526

a)

Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Total
86	10	21	218	30	161	526

b)

Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Total
86	10	21	213	30	166	526

c)

Constatou-se também que quanto mais perto se está da melhor solução, menores são as perdas nas linhas, desde 1,9484 MW, o valor mais alto, até 0,8281 MW, o valor mais baixo, que corresponde à melhor partícula (Figuras 4.11 e 4.12).

Na figura 4.12 aplicaram-se a esta rede os valores nodais da partícula que apresenta o valor de perdas mais elevado, que sabemos não ser a melhor solução para este caso de rede. Partícula da tabela 4.6.

Tabela 4.6: Uma partícula que não é a melhor solução

Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Total
26	6	10	86	15	39	182

Como se pode ver na figura 4.12 os dois geradores do barramento de referência/balanço já não estão perto de 0, apresentando um valor de potência necessário para fechar o balanço de potência da rede. Comparando este caso de rede com o da figura 4.11, pode-se ver que na maioria das linhas, os valores da taxa de ocupação são mais elevados do que os da figura 4.11 justificando por isso o valor de perdas mais elevado (1,9484 MW contra 0,8281 MW da figura 4.11).

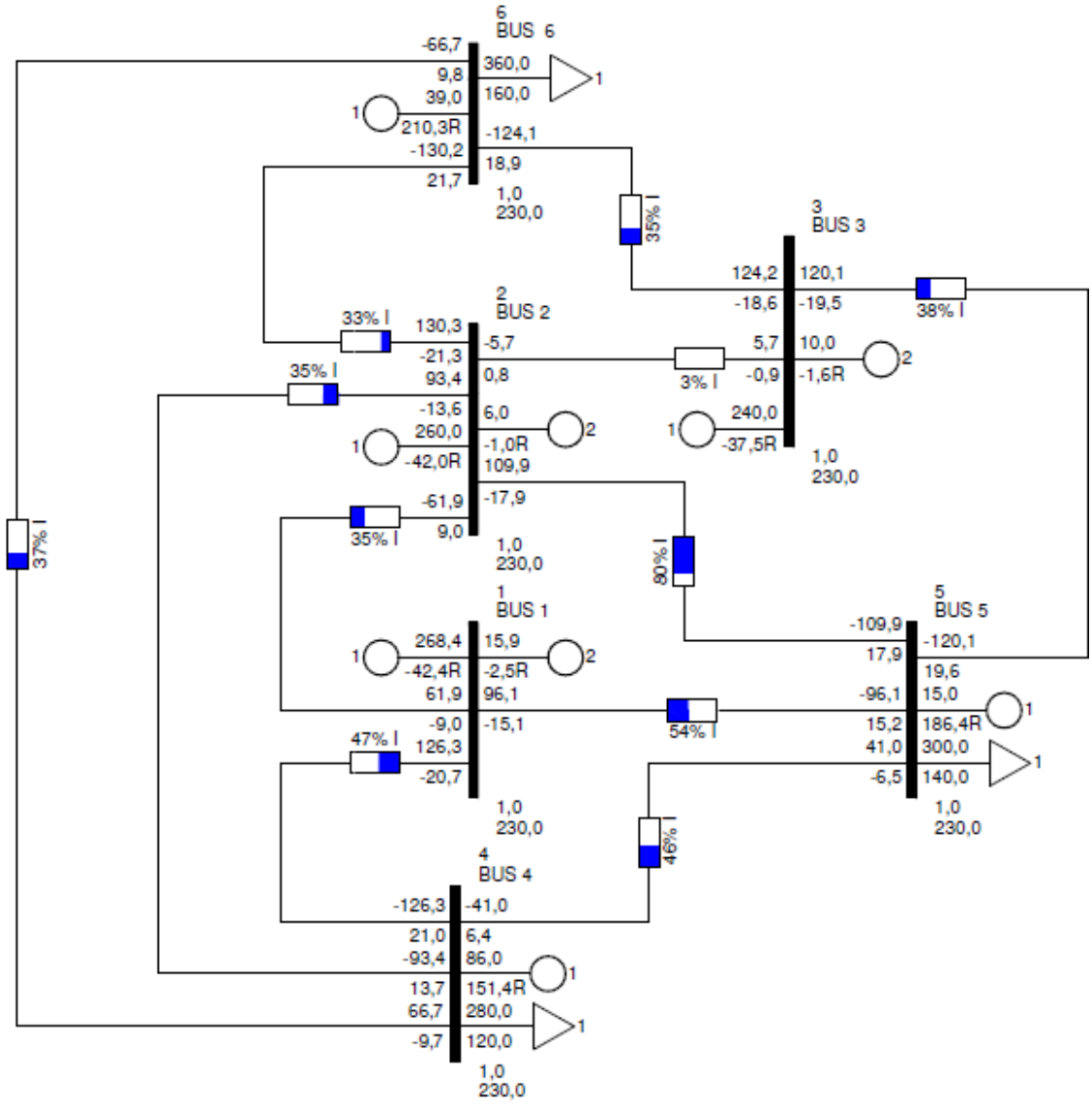


Figura 4.12: Rede de 6 barramentos, quando aplicados os valores de potência nodal de uma (não a melhor) partícula (tabela 4.6), mostrando a taxa de ocupação das linhas

4.3.2.2. Rede de 14 barramentos

O mesmo algoritmo PSO para injeções simultâneas, vai ser agora aplicado à rede de teste de 14 barramentos da figura 4.8.

Tal como se fez para a rede de 6 barramentos, vão também ser usados os valores de potência ativa máxima determinados em 4.3.1.2, tabela 4.8, para os vários valores de P_{max} nos vários nós da rede. Para todos os nós, será sempre $P_{min} = 0$.

À semelhança do que se fez na rede de 6 barramentos, apresenta-se também aqui (figura 4.13) o gráfico com o andamento da melhor partícula e da média de todas as partículas, em cada iteração. Os comentários ao gráfico da figura 4.16 são idênticos ao da figura 4.10 do ponto 4.3.2.1, sendo as diferenças as seguintes: o critério de paragem é o acontecimento do mesmo valor 7 vezes seguidas, porque devido à maior dimensão (mais do dobro) do espaço de procura, poderiam ser exigidas, sem necessidade, demasiadas iterações até ser encontrada a série de 10. Nos ensaios feitos, verificou-se que este critério é suficiente para o algoritmo encontrar a melhor solução. Pela mesma razão, dimensão do espaço de procura, aumentou-se a população para 50 partículas e a velocidade para 25. A lomba que se vê no gráfico nas iterações iniciais deve-se ao facto de devido à alta velocidade inicial, rapidamente ser alcançada uma ou mais iterações em que todas as partículas violam pelo menos uma restrição, sendo os seus valores nodais diminuídos de 5%, resultando que a média das partículas de cada iteração, cujo cálculo não tem em conta esta diminuição de valor, possa, durante algumas iterações, ser superior ao valor da melhor partícula.

Com o acumular de iterações e consequente diminuição da velocidade, devido ao fator de inércia dinâmico (ver ponto 4.2) e aprendizagem das partículas, este efeito rapidamente desaparece.

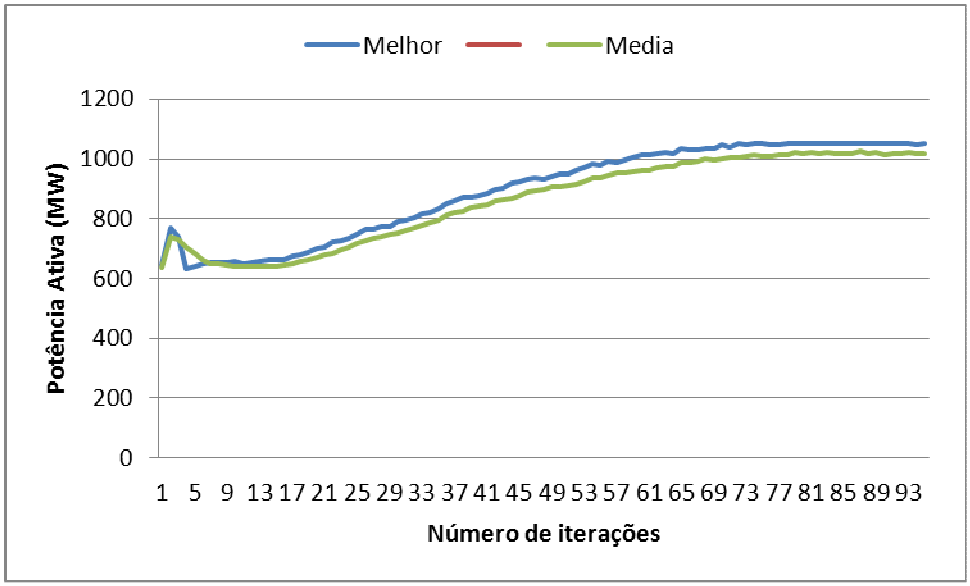


Figura 4.13: Progressão da melhor partícula e da média de todas as partículas com o número de iterações.

O máximo valor para o somatório das potências nodais da melhor ou das melhores partículas é de 1051 MW, como era esperado, que é o valor máximo de potência ativa que podemos injetar no nó de referência (tabela 4.7).

Foram aplicados os valores da melhor partícula (tabela 4.7) encontrada pelo algoritmo, à rede de 14 barramentos. Na figura 4.14 pode-se ver que o barramento de balanço, o 1, tem um valor muito próximo de 0, confirmando que esta é a melhor distribuição das potências ativas nodais, dentro da resolução definida (1 MW), para este caso de rede.

Tabela 4.7: A melhor partícula encontrada pelo algoritmo *PSO* quando aplicado à rede da figura 4.14

Nó	**1**	**2**	**3**	**4**	**5**	**6**	**7**
P_{max} (MW)	480	25	52	44	49	57	34
Nó	**8**	**9**	**10**	**11**	**12**	**13**	**14**
P_{max} (MW)	27	31	48	66	62	44	32

Como se pode ver na figura 4.14, a aplicação dos valores de potência ativa nodais desta partícula à rede de 14 barramentos, depois de feito o balanço de potência, faz com que o nó de balanço, o 1, fique com um valor muito próximo de zero. Não é zero porque o problema a resolver é discreto, o passo definido para o valor da potência é de 1 MW, não é uma variação contínua. Para este problema e para a resolução definida, pode-se dizer que este valor, 1051 MW (somatório das potências nodais da melhor partícula) é o valor máximo de potência ativa que pode ser injetado nos vários nós da rede, sem levar o nó de balanço para valores de potência ativa negativos. O valor seguinte, 1052 MW (tabela 4.8) já vai implicar uma potência negativa no nó de balanço, o que na prática equivale a ser instalada uma carga adicional nesse nó o que já seria outro cenário e outro caso de rede.

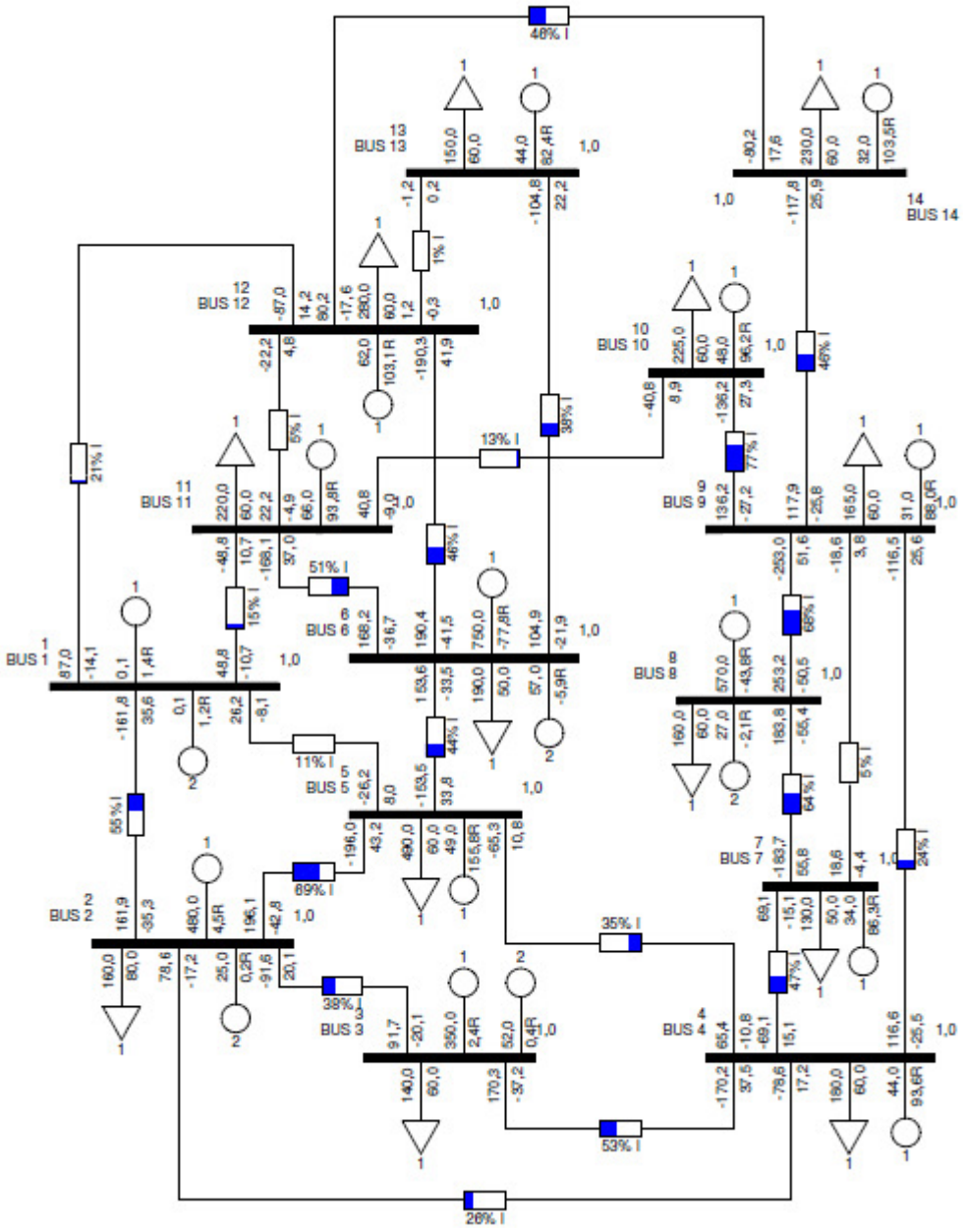


Figura 4.14: Rede de 14 barramentos, quando aplicados os valores de potência nodal da melhor partícula, mostrando a taxa de ocupação das linhas

Vão ser agora aplicados os valores da partícula da tabela 4.8 à rede de 14 barramentos. O resultado está representado na figura 4.15.

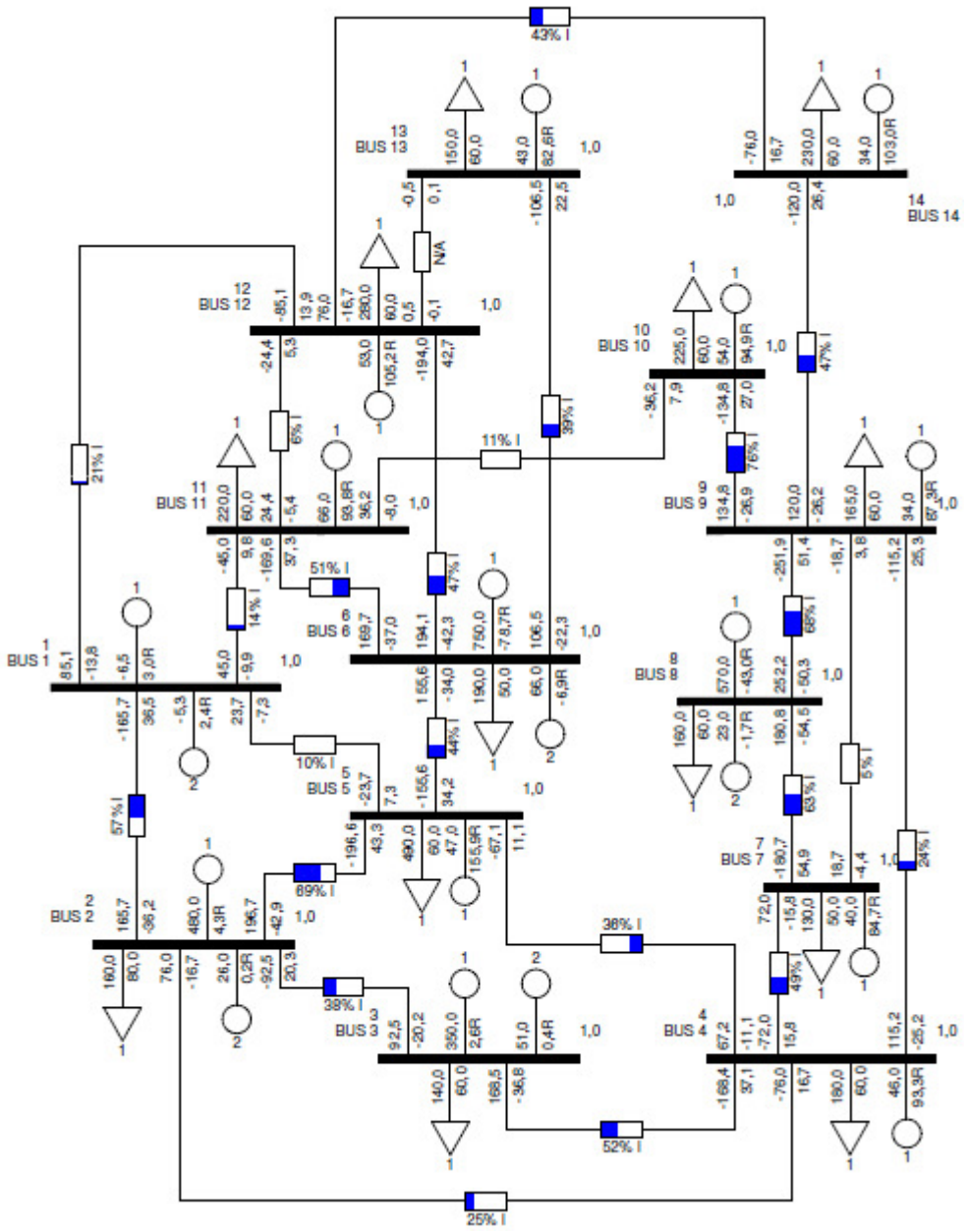


Figura 4.15: Rede de 14 barramentos, quando aplicados os valores de potência nodal da partícula da tabela 4.8, mostrando o valor de potência negativa no nó de balanço e a taxa de ocupação das linhas

Pode-se ver na figura 4.15 que o barramento de balanço tem um valor de potência ativa negativo (-11,8 MW), no conjunto dos dois geradores, o que viola uma das restrições impostas. O valor de potência ativa no barramento de balanço tem de ser positivo.

Tabela 4.8: Partícula encontrada pelo algoritmo *PSO*, quando aplicado à rede da figura 4.15, que provoca uma potência ativa negativa no nó de Balanço. O somatório das potências nodais desta partícula é 1052 MW

Nó	**1**	**2**	**3**	**4**	**5**	**6**	**7**
P_{max} (MW)	469	26	51	46	47	66	40
Nó	**8**	**9**	**10**	**11**	**12**	**13**	**14**
P_{max} (MW)	23	34	54	66	53	43	34

À semelhança do que aconteceu na rede de 6 barramentos, também aqui se obtiveram mais do que uma partícula com o mesmo somatório de valores de potência ativa nodal, embora os valores em alguns dos nós da rede sejam diferentes. Mais uma vez, o valor das perdas nas linhas, em cada caso, pode ser um fator de decisão para o gestor da rede. A partícula da tabela 4.9, por exemplo, tem o somatório dos valores da potência ativa nodal igual ao da melhor partícula (1051 MW), mas apresenta perdas de 5,2669 MW. As perdas da melhor partícula são de 5,2319 MW.

Tabela 4.9: Partícula com o somatório dos valores de potência ativa nodal igual ao da melhor partícula, mas com maiores perdas nas linhas

Nó	**1**	**2**	**3**	**4**	**5**	**6**	**7**
P_{max} (MW)	480	26	52	45	50	55	38
Nó	**8**	**9**	**10**	**11**	**12**	**13**	**14**
P_{max} (MW)	28	31	47	66	58	43	32

4.3.2.3. Análise dos resultados

No caso da rede de 6 barramentos o algoritmo *PSO* encontra a melhor, ou as melhores partículas, muito mais rapidamente do que no caso da de 14 (29 iterações contra 94), como era esperado, visto haver, com 14 valores, muito mais combinações possíveis do que com 6. Por isso no caso da rede de 14 número de partículas foi aumentado de 30 para 50.

Tanto num caso como no outro, se pode ver que o somatório dos valores nodais de potência ativa, apresentado pelas melhores partículas (526 MW no caso da rede de 6 nós e 1051 MW no caso da rede de 14 nós) é igual ao valor máximo de potência ativa que se pode injetar no nó de balanço. Ambas as redes, como apresentadas inicialmente, resolvendo o trânsito de energia, ficam com esses valores no nó de balanço. Isto acontece, porque com a topologia destas redes e a geração instalada essa é a distribuição de potências necessária para alimentar as cargas instaladas. A ser injetada potência ativa, em quaisquer nós da rede, a potência injetada vai sendo diminuída no nó de referência (é este o responsável por fechar o balanço energético), porque se as cargas instaladas são as mesmas, a potência necessária para as alimentar também tem de ser a mesma.

Para ilustrar isto, foram apresentados dois exemplos:

1. Com a rede de 6 barramentos, onde “foi injetada uma partícula” cujo somatório de potências ativas nodais não é suficiente para alimentar as cargas instaladas (tabela 4.6) e o resultado do trânsito de energia foi que o nó de balanço teve de compensar

essa falta injetando potência ativa na rede. Veja-se a potência ativa positiva em ambos os geradores do nó de balanço (figura 4.12).

2. Com a rede de 14 barramentos, onde "se injetou uma partícula" cujo somatório de potências ativas nodais era superior ao valor necessário para alimentar as cargas instaladas (tabela 4.8). O que aconteceu foi que, resolvido o trânsito de energia, o nó de balanço teve de fazer a compensação dessa potência ativa em excesso, tomando um valor negativo, em ambos os seus dois geradores, como se pode ver na figura 4.15. Sabendo que uma potência ativa negativa num nó significa que está a ser injetada potência nesse nó, que esse nó está a consumir energia, isto equivale a dizer que para fechar o balanço de potência o nó de balanço teve de instalar mais carga na rede, porque numa rede de energia elétrica não pode transitar mais potência do que aquela necessária para alimentar as cargas instaladas mais a que é perdida nas linhas.

Estes dois exemplos ilustram o objetivo da aplicação do algoritmo *PSO* a estes dois casos de rede e mostram que esse objetivo foi alcançado, ou seja, o algoritmo conseguiu encontrar as melhores partículas (melhores configurações de potências nodais), de modo a maximizar a potência ativa injetada na rede.

Verificou-se também que em ambos as redes, 6 e 14 nós, o algoritmo encontra várias partículas com o mesmo valor de somatório das potências nodais. Como forma de "desempate" entre estas partículas,

foram usadas as perdas nas linhas que cada caso apresenta e considerada como melhor partícula aquela que apresenta menos perdas.

Este resultado, de mais do que uma solução (partícula) com o máximo de potência ativa total a injetar na rede, mas com perdas totais nas linhas diferentes, que acontece em ambas as redes de teste, só por si pode já justificar uma análise da rede, porque proporciona uma maior flexibilidade na escolha da melhor configuração das injeções, de modo a que a rede absorva o acréscimo de energia sem, ou com o mínimo de reforço da sua estrutura.

5

Quinto Capítulo - **Conclusão**

5.1. **Observações finais**

No início desta dissertação foi proposto resolver-se o problema da máxima injeção nodal, simultânea e não simultânea, com a ajuda do algoritmo *PSO*. Escolheu-se este tema porque se considerou importante a determinação da máxima potência ativa que se pode injetar em cada nó da rede, de modo a poder-se planeá-la e operá-la com eficiência. Considerou-se esta determinação tanto mais importante quanto maior é o número de gerações de pequena ou média dimensão e dependentes de fatores geográficos como é o caso da maioria das fontes renováveis.

No capítulo 3 o algoritmo *PSO* foi apresentado, foi adaptado ao problema a resolver e implementado, senso justificada a sua escolha.

No capítulo 4 o algoritmo *PSO* foi aplicado a duas redes, uma de 6 e outra de 14 barramentos, baseadas nas redes de teste *IEEE 6*

BUS e *IEEE 14 BUS*, respetivamente, para o caso das injeções simultâneas. No caso das injeções não simultâneas, dada a menor complexidade do problema foi utilizado um *algoritmo de busca Gaussiana*, aplicado às mesmas duas redes.

Devido à forma como o algoritmo *PSO* procura a solução, várias partículas que evoluem no espaço de *d* dimensões (sendo *d* igual ao número de barramentos da rede), com valores de posição e de velocidade diferentes, embora haja uma convergência para o melhor valor de somatório das potências nodais, são encontradas várias partículas que tendo esse valor como somatório, apresentam valores de potência ativa, em cada nó, ligeiramente diferentes umas das outras. Foi acrescentado ao algoritmo a funcionalidade de calcular as perdas nas linhas em cada um desses casos para se disponibilizar mais informação ao gestor da rede.

Feitos estes ensaios e analisados os resultados, conclui-se que o algoritmo *PSO* é uma boa solução técnica para estas determinações, sendo os seus parâmetros escolhidos de acordo com a especificidade de cada rede para maximizar a qualidade dos resultados obtidos. No caso em estudo, o que faz a diferença entre a rede de 6 barramentos e a de 14 é haver, na de 14 nós, muito mais combinações, ou seja, muito mais soluções possíveis. Devido a isso, ao passar da rede de 6 nós para a de 14, foram alterados os parâmetros *velocidade máxima da partícula*, que passou de 10 para 25 e o *número de partículas* que passou de 30 para 50.

5.2. Perspetivas de desenvolvimento futuro

Vão ser agora apresentadas algumas sugestões para desenvolvimento futuro, de modo a complementar o estudo realizado:

1. Incluir um módulo, no software, que faça a gestão do ponto de vista económico, das injeções nodais, ou seja, sempre que se injeta potência num nó, vá diminuindo a potência injetada nos outros nós, começando pelas centrais de produção mais cara, caso seja aplicável. Por exemplo, reduzir geração por ordem de mérito adaptado ao cenário de rede em causa.

2. Implementar diferentes variações do algoritmo *PSO* para comparação com a utilizada. Por exemplo, *PSO* with Constriction Factor (CFPSO) [25], Comprehensive Learning *PSO* (CLPSO) [25] ou aplicar a técnica de Path Relinking [26].

3. Incluir uma funcionalidade que permita fazer este estudo, da máxima injeção nodal, numa determinada zona da rede em vez de em toda a rede. Esta funcionalidade só faz sentido para o caso das injeções simultâneas.

4. Aplicar o algoritmo *PSO* a diferentes cenários de carga/geração, como verão e inverno, vazio e ponta e inferir sobre a sua parametrização.

Referências Bibliográficas

[1] Rau N.S. e Wan Y.H.: *"Optimum location of resources in distributed planning"*, IEEE Trans. Power. Syst., 1994, 9, (4), pp. 2014 - 2020.

[2] Kim K.H., Lee Y.J., Rhee S.B., Lee S.K. e You S.K.: *"Dispersed generator placement using fuzzy-GA in distribution systems"*. Proceedings IEEE Power Engineering Society Summer Meeting, Chicago, Il, 21-25 July 2002, pp. 1148-1153.

[3] Harrison G.P. e Wallace A.R. (2005), *"Optimal Power Flow Evaluation of Distribution Network Capacity for the connection of Distributed Generation"*, Proceedings Institute Electric Engineers - Generation, Transmission and Distribution, Vol. 152, no. 1 Janeiro 2005, páginas 115-122.

[4] Nunes, J.G., *"Desenvolvimento de algoritmos para a determinação da máxima injeção nodal em redes de energia elétrica"*, ISEL, Dissertação de mestrado em Engenharia Eletrotécnica, ramo de Energia, Setembro 2012.

[5] Pires B.A., *"Maximização da Penetração da Geração Distribuída Através do Algoritmo de Otimização Nuvem de Partículas"*, Univer-

sidade Federal do Rio Grande do Norte, programa de Pós Graduação Elétrica e de Computação, Agosto 2011.

[6] Almeida J.M., *"Avaliação Probabilística da Capacidade de Receção Nodal de uma Rede de Transporte"*, Dissertação de Mestrado Integrado em Engenharia Eletrotécnica e de Computadores, FEUP, Julho 2012.

[7] White T. e Pagurek B. *"Tuwards Multi-Swarm Problem Solving in Networks"*, Proceedings of the 3rd International Conference on Multi-Agent Sysytems (ICMAS'98), p. 333-340, July, 1998.

[8] Heppner F. e Grenander U., *"A Stochastic Nonlinear Model for Coordinated Bird Flocks"*, The Ubiquity of Caos, AAAS Publications, Washing, ton DC, 1990.

[9] Kennedy J. e Eberhart R., *"Particle Swarm optimization"*, Proceedings of the IEEE International Conference on Neural Networks, p. 1942-1948, New Jersey, USA 1995.

[10] Rosendo M., *"Um Algoritmo de Otimização por Nuvem de Partículas para Resolução de Problemas Combinatórios"*, Dissertação apresentada ao programa de Pós-Graduação em Informática, Setor de Ciências Exatas, como requisito parcial à obtenção de grau de Mestre, Universidade Federal do Paraná, Curitiba, 2010.

[11] Viveros R.C., *"Ajuste Coordenado de Controladores de Sistemas de Potência usando Metaheurísticas"*, Tese Dsc, UFRJ-COPPE, 2007.

[12] Abido M.A., *"Optimal Design of Power System Stabilizers Using Particle Swarm optimization"*, Energy Conversion, IEEE Transactions on vol 17, nº 3, p. 406-413, Setembro 2002.

[13] Berg F. e Engelbrecht A.P., *"A study of particle swarm optimization particle trajectories"*, Department of Computer Science, University of Pretoria, Roperstreet, Pretoria 0002, South Africa, Fevereiro 2005.

[14] Paiva J.P., *"Redes de Energia Eléctrica, Uma Análise Sistémica"*, 3ª Edição (revista), IST Press, Agosto de 2011, ISBN: 978-989-8481-06-1.

[15] Siemens USA. Smart grid solutions http://w3.usa.siemens.com/smartgrid/us/en/transmission-grid/products/grid-analysis-tools/transmission-system-planning/Pages/University-Order.aspx, *"PSSE University order form"*, Janeiro 2014.

[16] Python Software Foundation (US) https://www.python.org/downloads/release/python-279/,"*python*, Fevereiro 2014.

[17] New Mexico Tech Computer Center http://infohost.nmt.edu/tcc/help/pubs/python/web/index.html, *"Python 2.7 quick reference"*, Fevereiro 2014.

[18] Python Software Foundation (US) https://docs.python.org/2/reference, *"The Python Language Reference"*, Fevereiro 2014.

[19] whit. Python for Power Systems https://psspy.org/psse-help-forum, Fevereiro - Outubro 2014.

[20] Eclipse Foundation http://www.eclipse.org/downloads, *"Eclipse"*, Fevereiro 2014.

[21] PyDev http://pydev.org/download.html, *"Pydev for Eclipse"*, http://pydev.org/manual_101_root.html, *"Pydev installation instructtions"*, Fevereiro 2014.

[22] Shi Y. e Eberhart R.C., *"Parameter Selection in Particle Swarm Optimization"*, Lecture Notes in Computer Science; Vol. 1447, Proceedings of the 7th International Conference on Evolutioary Programming VII pp. 591 - 600, 1998.

[23] Wood A.J. e Wollenberg B.F., *"Power Generation, Operation and Control"*, 2ª Edição, John Wiley & Sons, Inc., 1996, ISBN: 0-471-58699-4.

[24] Dabbagchi I. e Christie R., *"Power System Test Case Archive"*, University of Washington, http://www.ee.washington.edu/research/pstca, 2014.

[25] Ganguly S., Sahoo N.C. e Das D., *"Mono- and multi-objective planning of electrical distribution networks using particle swarm optimization"*, Department of Electrical Engineering, Indian Institute of Technology, Kharagpur 721302, India, Outubro de 2010.

[26] Glover F., Laguna M. e Martí R., *"Fundamentals of Scatter Search and Path Relinking"*, Graduate School of Business and Administration, University of Colorado, Boulder, CO 80309-0419, USA, {Fred.Glover}{Manuel.Laguna}@Colorado.edu.
Departamente de Estadística e Investigación Operativa, Facultad de Matemáticas, Universidad de Valência, Dr. Moliner 50, 46100 Burjassot (Valencia) Spain, Rafael.Marti@uv.es.

Anexo I - Dados das Redes de Teste

Rede de 6 barramentos

Dados dos barramentos - geração e carga:

Barramento		Tensão	Geração		Carga	
Nº	Tipo	(pu)	P (MW)	Q (MVar)	P (MW)	Q (MVar)
1	Referência	1,0000	440	41	0	0
2	PV	1,0000	260	230	0	0
3	PV	1,0000	150	150	0	0
4	PQ	0,9986	0	0	280	120
5	PQ	0,9989	0	0	300	140
6	PQ	0,9981	0	0	360	160

Dados das linhas:

Linhas	Resistência (pu)	Reatância (pu)	Suscetância (pu)
1 - 2	0,000263	0,001799	0,000631
1 - 4	0,000363	0,002199	0,000831
1 - 5	0,000363	0,002299	0,000931
2 - 3	0,000263	0,001799	0,000631
2 - 4	0,000263	0,001799	0,000631
2 - 5	0,000163	0,001000	0,000431
2 - 6	0,000363	0,002199	0,000731
3 - 5	0,000163	0,001000	0,000431
3 - 6	0,000363	0,002399	0,000731
4 - 5	0,000220	0,001400	0,000531
4 - 6	0,000263	0,001799	0,000631

Rede de 14 barramentos:

Dados dos barramentos - geração e carga:

Barramento		Tensão	Geração		Carga	
Nº	Tipo	(pu)	P (MW)	Q (MVar)	P (MW)	Q (MVar)
1	Referência	1,000	571	109,17	0	0
2	PV	1,000	480	151,70	160	80
3	PV	1,000	350	93,84	140	60
4	PQ	0,9990	0	0	180	60
5	PQ	0,9993	0	0	490	60
6	PV	1,000	750	280,64	190	50
7	PQ	0,9988	0	0	130	50
8	PV	1,000	570	149,19	160	60
9	PQ	0,9984	0	0	165	60
10	PQ	0,9980	0	0	225	60
11	PQ	0,9989	0	0	220	60
12	PQ	0,9990	0	0	280	60
13	PQ	0,9986	0	0	150	60
14	PQ	0,9978	0	0	230	60

Dados das linhas:

Linhas	Resistêcia (pu)	Reatância (pu)	Suscetância (pu)
1 - 2	0,000263	0,001200	0,000631
1 - 5	0,000463	0,001500	0,000731
1 - 11	0,000263	0,001200	0,000631
1 - 12	0,000163	0,001000	0,000200
2 - 3	0,000263	0,001200	0,000631
2 - 4	0,000263	0,001200	0,000631
2 - 5	0,000263	0,001200	0,000631
3 - 4	0,000263	0,001200	0,000631
4 - 5	0,000363	0.002200	0,000431
4 -7	0,000263	0,001200	0,000631
4 - 9	0,000263	0,001200	0,000631
5 - 6	0,000263	0,001200	0,000631
6 - 11	0,000263	0,001200	0,000631
6 - 12	0,000263	0,001200	0,000631
6 - 13	0,000463	0,002200	0,000231
7 - 8	0,000363	0,001200	0,000631
7 - 9	0,000263	0,012000	0,006310
8 - 9	0,000363	0,001800	0,000631

Dados das linhas (continuação):

Linhas	Resistência (pu)	Reatância (pu)	Suscetância (pu)
9 - 10	0,000100	0,000500	0,000231
9 - 14	0,000263	0,001200	0,000631
10 - 11	0,000263	0,001200	0,000631
11 - 12	0,000263	0,001200	0,000631
12 - 13	0,000263	0,001200	0,000631
12 - 14	0,000263	0,001200	0,000631

Anexo II - Exemplo de Código Escrito Em Linguagem Python

Como exemplo do código desenvolvido, apresenta-se abaixo um dos módulos da aplicação. O *Module PSO*.

Como já se disse no Quarto Capítulo - Aplicação, é este módulo que implementa o algoritmo PSO.

Na nossa aplicação utilizámos as seguintes 4 funções:

f_create_swarm_1

f_update_swarm_1

f_evaluate_particule

f_set_best_particle

```
'''
Created on 26/09/2014

@author: Luis_2
'''

from __future__ import print_function
import random

# INICIA O ENXAME DE PARTICULAS s, COM VALORES ALEATORIOS, SO NOS BUS ONDE JA
# HA GERACAO. oS OUTROS FICAM A 0
# INICIA O VECTOR DE MELHORES POSICOES DAS PARTICULAS (pbest) COM VALORES
# IGUAIS AOS DE s
def f_create_swarm(num_p, size_p, min_power, max_power, v_min, v_max, aux):

    size_aux = size_p
    size_p *= 4

    # Enxame - num_p de particulas de size_p dimensoes
    s = [size_p*[0] for i in range(num_p)]
    # Primeira parte de cada s[i] -> posicao da particula. Segunda -> pbest.
    # Terceira -> velocidade.
    # Quarta -> perdas totais, potencia total, lista das potencias minimas e
    # lista das potencias maximas, por esta ordem
    # Inicia as particulas, as suas melhores posicoes com valores aleatórios
    # entre min_power e max_power
    # e as suas velocidades com valores aleatorios entre v_min e v_max
    for i in range(num_p):
        for j in range(size_p/4):
            # Se e barramento com gerador
            if aux[j] == True:
                # Inicia a primeira parte do vetor (posicao da particula)
                s[i][j] = random.randint(min_power, max_power)
                # Inicia a segunda parte do vetor (pbest da partícula).
                # No inicio, pbest = posicao
                s[i][j+size_p/4] = s[i][j]
                # Inicia a terceira parte do vetor (velocidade da particula)
                s[i][j+(size_p/4)*2] = random.randint(v_min, v_max)
        # Inicia as perdas com max_power, para que na primeira iteracao do
        # Main as perdas sejam menores e haja actualizacao
        s[i][size_aux*3] = max_power

    return s


# INICIA O ENXAME DE PARTICULAS s, COM VALORES ALEATORIOS
# INICIA O VECTOR DE MELHORES POSICOES DAS PARTICULAS (pbest) COM VALORES
# IGUAIS AOS DE s
def f_create_swarm_1(num_p, size_p, min_power, max_power, ipot, v_min,v_max):

    # Enxame - num_p particulas de size_p dimensoes
    s = [(size_p*4)*[0] for i in range (num_p)]
    # Primeira parte de cada s[i] -> posicao da particula.
    # Segunda -> pbest. Terceira -> velocidade.
    # Quarta -> perdas totais, potencia total, sinalizacao de violacao de
    # restricao (por esta ordem)
    # Inicia as particulas, as suas melhores posicoes com valores aleatorios
    # e as suas velocidades com valores aleatorios entre v_min e v_max
```

```
    for i in range(num_p):
        for j in range(size_p):
            # Inicia a primeira parte do vetor (posicao da particula)
            s[i][j] = random.randint(min_power[j], max_power[j]/ipot)
            # Inicia a segunda parte do vetor (pbest da particula).
            # No inicio, pbest = posicao
            s[i][j+size_p] = s[i][j]
            # Inicia a terceira parte do vetor (velocidade da particula)
            s[i][j+size_p*2] = random.randint(v_min, v_max)
            # Inicialmente a potencia total (somatorio das potencias
            # atribuidas aos bus) fica a 0
        # No inicio nao ha violacao de restricoes
        s[i][size_p*3+2] = False
        # Inicia as perdas com max_power, para que na primeira iteracao
        # do Main as perdas sejam menores e haja actualizacao
        s[i][size_p*3+2] = max_power[1]

    return s


# ACTUALIZA A VELOCIDADE E A POSICAO DE CADA PARTICULA
# NOS BUS COM GERACAO  ORIGINAL
def f_update_swarm(s, k, k_max, w_min, w_max, v_min, v_max, gbest, num_p,
                   size_p, min_power, max_power, aux):

    # Componentes individual (c1) e social (c2)
    c1 = c2 = 1.49618

    # Calcula o novo factor de inercia
    w = w_max - ((w_max - w_min)/k_max)*k

    # Atualiza a velocidade de cada particula
    for i in range(num_p):
        for j in range(size_p):
            # Se e barramento com gerador
            if aux[j] == True:
                # Nova velocidade: w*v_anterior -> inercia x velocidade
                # da iteracao anterior
                s[i][j+size_p*2] = w * s[i][j+size_p*2]
                # Nova tendencia individual: v + (c2 * r2 * (pbest-xi))
                s[i][j+size_p*2] = s[i][j+size_p*2] +
c1*random.random()*(s[i][j+size_p] - s[i][j])
                # Nova tendencia social: v + (c2 * r2 * (gbest-xi))
                s[i][j+size_p*2] = s[i][j+size_p*2] +
c2*random.random()*(gbest[j] - s[i][j])
                # Garante que esta dentro dos limites de velocidade
                if s[i][j+size_p*2] < v_min:
                    s[i][j+size_p*2] = v_min
                else:
                    if s[i][j+size_p*2] > v_max:
                        s[i][j+size_p*2] = v_max

    # Atualiza a posicao de cada particula
    for i in range(num_p):
        for j in range(size_p):
            if aux[j] == True:
                s[i][j] = s[i][j] + int(round(s[i][j+size_p*2]))
                if s[i][j] > max_power:
```

```
                    s[i][j] = max_power
                if s[i][j] < min_power:
                    s[i][j] = min_power

    return s


# ACTUALIZA E VELOCIDADE E A POSICAO DE CADA PARTICULA, EM TODA A REDE
def f_update_swarm_1(s, k, k_max, w_min, w_max, v_min, v_max, gbest, num_p,
                     size_p, min_power, max_power):

    c1 = c2 = 1.49618    # Componentes individual (c1) e social (c2)

    # Calcula o novo factor de inercia
    w = w_max - ((w_max - w_min)/k_max)*k

    # Atualiza a velocidade de cada particula
    for i in range(num_p):
        # Salva o somatorio anterior
        s[i][size_p*3+3] = s[i][size_p*3+1]
        # Reset ao somatorio das potencias
        s[i][size_p*3+1] = 0

        for j in range(size_p):
            # Nova velocidade: w*v_anterior -> inercia x velocidade
            # da itera  cao anterior
            s[i][j+size_p*2] = int(round(w * s[i][j+size_p*2]))
            # Nova tendencia individual: v + (c2 * r2 * (pbest-xi))
            s[i][j+size_p*2] = s[i][j+size_p*2] +
int(round(c1*random.random()*(s[i][j+size_p] - s[i][j])))
            # Nova tendencia social: v + (c2 * r2 * (gbest-xi))
            s[i][j+size_p*2] = int(round(s[i][j+size_p*2])) +
int(round(c2*random.random()*(gbest[j] - s[i][j])))
            # Garante que esta dentro dos limites de velocidade
            if s[i][j+size_p*2] < v_min:
                s[i][j+size_p*2] = v_min
            else:
                if s[i][j+size_p*2] > v_max:
                    s[i][j+size_p*2] = v_max
            # Atualiza a posicao da particula
            s[i][j] += s[i][j+size_p*2]
            # Garante que esta dentro dos limites de potencia
            if s[i][j] < min_power[j]:
                s[i][j] = min_power[j]
            if s[i][j] > max_power[j]:
                s[i][j] = max_power[j]
            # Actualiza o somatorio das potencias
            s[i][size_p*3+1] += s[i][j]

        # Reset ao sinal de violacao de restricao para inicir nova iteracao
        s[i][size_p*3+2] = False

    return s
```

```
# AVALIA CADA PARTICULA (perdas)
def f_evaluate_particule_p(s, i, l, k, K_best_p, size_p):

    print(l[0])
    print(' ')
    print('k['+ str(k) +']'+'s['+ str(i) +'] -> '+ str(s[i]))
    print(' ')
    # Compara a particula com ela propria, da iteracaoa K anterior
    # PBEST - Se as perdas da particula s[i] < do que as que la estao,
    # actualiza as perdas e pbest
    if l[0] < s[i][size_p*3]:
        # Actualiza pbest da particula
        for j in range(size_p):
            s[i][j+size_p] = s[i][j]
            # Actualiza perdas da particula
            s[i][size_p*3] = l[0]

    # A MELHOR PARTICULA DESTA ITERACACAO K
    # Se as perdas correntes < que as menores ate agora nesta iteracao k
    if l[0] < K_best_p[1]:
    # Salva indice da melhor particula (menores perdas) ate agora,
    # nesta iteracao k
        K_best_p[0] = i
        # Salva menores perdas ate agora, nesta iteracao k
        K_best_p[1] = l[0]

    return K_best_p


# AVALIA CADA PARTICULA (maxima injeccao nodal simultanea)
def f_evaluate_particule(s, i, k, k_best, size_p):

    # Compara o somatorio das potencias com o da iteracao K anterior
    # PBEST - Se a particula s[i] e melhor que a da iteracao anterior
    # (somatorio das potencias >)
    # e a particula nao violou nenhuma restricao actualiza pbest
    if s[i][size_p*3+1] > s[i][size_p*3+3] and s[i][size_p*3+2] == False:
        for j in range(size_p):            # Actualiza pbest
            s[i][j+size_p] = s[i][j]

    # A MELHOR PARTICULA DESTA ITERACACAO K
    # Se o somatorio das potencias nodais desta particula > que o maior
    # ate agora nesta iteracao k e a particula nao violou nenhuma restricao
    if s[i][size_p*3+1] > k_best[1] and s[i][size_p*3+2] == False:
        # Salva indice da melhor particula (maior somatorio das potencias
        # nodais) ate agora, nesta iteracao k
        k_best[0] = i
        # Salva o maior somatorio das potencias nodais ate agora,
        # nesta iteracao k
        k_best[1] = s[i][size_p*3+1]
        k_best[2] = False

    return k_best
```

```
# A MELHOR PARTICULA DE TODAS AS  ITERACOES (perdas)
def f_set_best_particle_p(s, k, size_p, best_p, k_best_p, gbest,
best_particle):

    # GBEST - Compara as perdas desta particula com as menores perdas
    # ate agora. Se forem menores, actualiza best
    if k_best_p[1] < best_p[1]:
        # Menores perdas ate agora
        best_p[1] = k_best_p[1]
        # Indice da melhor particula (menores perdas) ate agora
        best_p[0] = k_best_p[0]
        # Salva a iteracao onde encontrou a melhor particula ate agora
        best_p[2] = k
        for n in range(size_p*4):
            # Salva a melhor particula ate agora
            best_particle[n] = s[k_best_p[0]][n]
            # E actualiza gbest
        for j in range(size_p):
            gbest[j] = s[k_best_p[0]][j]

    return best_p


# A MELHOR PARTICULA DE TODAS AS ITERACOES
# (maxima injeccao nodal simultanea)
def f_set_best_particle(s, k, size_p, best, k_best, gbest, best_particle):

    # GBEST - Compara o somatorio das potencias nodais desta particula
    # com o maior  ate agora. Se for maior, actualiza best
    if k_best[1] > best[1]:
        # Maior somatorio ate agora
        best[1] = k_best[1]
        # Indice da melhor particula (maior somatorio) ate agora
        best[0] = k_best[0]
        # Salva a iteracao onde encontrou a melhor particula ate agora
        best[2] = k
        # Salva a melhor particula ate agora
        for n in range(size_p*4):
            best_particle[n] = s[k_best[0]][n]
        # E actualiza gbest
        for j in range(size_p):
            gbest[j] = s[k_best[0]][j]

    return best
```